

# **Inktomi Enterprise Search Customization Guide**

**Version 4.3**

Publication Date: 12/10/01

## **Copyrights and Trademarks**

© Copyright 1999—2001 Inktomi Corporation. All Rights Reserved.

This document contains proprietary and confidential information of Inktomi Corporation. The contents of this document may not be disclosed to third parties, copied or duplicated in any form, in whole or in part, without the prior written permission of Inktomi Corporation.

INKTOMI and the tri-color cube design are trademarks of Inktomi Corporation.

### **RESTRICTED RIGHTS LEGEND**

Use, duplication, or disclosure of the technical data contained in this document by the Government is subject to restrictions as set forth in subdivision (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 52.227-7013 and/or in similar or successor clauses in the FAR, or in the DOD or NASA FAR Supplement. Unpublished rights reserved under the Copyright Laws of the United States. Contractor/manufacturer is Inktomi Corporation, 4100 East 3rd Avenue, Foster City, CA, 94404, U.S.A.



## Table of Contents

*Products*

<b>PREFACE</b>	<b>7</b>	<b>Preface</b>
	<b>8</b>	Navigating this document
	<b>9</b>	Who should read this book
	<b>10</b>	Related Reading
 <b>CHAPTER 1</b>	 <b>11</b>	 <b>Customizing with the Admin Interface</b>
	<b>12</b>	Using the Admin Interface
	<b>13</b>	Query parameters
	<b>15</b>	Displaying searchable collections at the query form
	<b>17</b>	CCE parameters
	<b>18</b>	CCE initial pages
	<b>18</b>	CCE topic browse pages
	<b>18</b>	CCE search pages
 <b>CHAPTER 2</b>	 <b>23</b>	 <b>Active Doc Directory Customizations</b>
	<b>24</b>	Understanding the document structure
	<b>25</b>	Changing headers and footers
	<b>26</b>	Tables
	<b>27</b>	Building initial search pages
	<b>27</b>	The indexform_.html series
	<b>28</b>	Topics
	<b>30</b>	Building search results pages
	<b>31</b>	The queryform_.html series
	<b>38</b>	Thesaurus
	<b>39</b>	Hits or no hits
	<b>41</b>	Search results page builder quick reference
	<b>43</b>	Building the help pages
	<b>43</b>	Help header and footer files
	<b>44</b>	Help content files
	<b>45</b>	Other files in the /help directory
	<b>46</b>	Hiding links under the More Services heading
	<b>47</b>	Getting ready to customize
	<b>47</b>	Protecting customized files during upgrades
	<b>47</b>	Creating a new active document directory
	<b>47</b>	Accessing content in the new directories
	<b>48</b>	Editing for customization
	<b>48</b>	Publishing changes
	<b>49</b>	Additional search presence
 <b>CHAPTER 3</b>	 <b>51</b>	 <b>Customizing with Form Variables</b>
	<b>52</b>	Changing the results page
	<b>53</b>	Search specification

	<b>53</b>	col
	<b>55</b>	qt
	<b>55</b>	nh
	<b>57</b>	st
	<b>57</b>	lk
	<b>58</b>	rf
	<b>58</b>	rq
	<b>59</b>	oq
	<b>60</b>	qp
	<b>61</b>	qs
	<b>62</b>	qp combined with qs
	<b>64</b>	pw
	<b>64</b>	ws
	<b>65</b>	ql
	<b>65</b>	fs
	<b>67</b>	ct
	<b>67</b>	ht
	<b>68</b>	si
	<b>68</b>	ex
<b>69</b>		Search box display variables
	<b>69</b>	qm
	<b>69</b>	qc
<b>71</b>		Advanced Search variables
	<b>71</b>	op
	<b>72</b>	fl
	<b>72</b>	ty
	<b>73</b>	tx
	<b>73</b>	dt
	<b>76</b>	la
<b>77</b>		Implementing search form variables
	<b>77</b>	Hidden form elements
	<b>77</b>	Integrating variables into checkboxes
	<b>78</b>	Integrating form variables into a listbox
	<b>79</b>	Integrating form variables into radio buttons
<b>CHAPTER 4</b>	<b>81</b>	<b>Integration</b>
	<b>82</b>	Style sheets
	<b>82</b>	Calling Inktomi Search Software's style sheet
	<b>82</b>	default.css
	<b>83</b>	sample.css
	<b>84</b>	Managing frame sets
	<b>84</b>	Sending results pages to a surrogate frame set
<b>CHAPTER 5</b>	<b>87</b>	<b>Changing the patches.py file</b>
	<b>88</b>	About Python and patches.py
	<b>91</b>	Monitoring changes
	<b>92</b>	Default values for new collections
<b>CHAPTER 6</b>	<b>95</b>	<b>Using Content Assistants</b>

- 96** Understanding Content Assistants
  - 96** Evaluating Documents
  - 96** Example Uses
- 97** Understanding the Service Provider Interface
  - 98** Content Assistant States

## **CHAPTER 7**

### **99 Customization Examples and Templates**

- 100** Customizing the user interface
  - 100** Adding a logo to every page
  - 100** Adding a logo to the search box
  - 101** Adding a search box to your home page
  - 101** Creating a stand-alone search form
  - 102** Limiting the collections displayed in the search box
  - 104** Displaying each collection name on a separate line
  - 105** Displaying two URLs per hit
  - 105** Adding an image to each search result
  - 108** Changing other content of the results page
  - 108** Creating a non-tables version of public pages
  - 109** Changing the rotating tips file
- 110** Customizing index behavior
  - 110** Adding URLs to the index
  - 110** Adding URLs automatically
  - 110** Changing the date sort
  - 110** Changing document titles
  - 111** Changing document summaries
  - 111** Returning ALL results for a particular query
  - 112** Indexing multiple fields for a single field search

## **APPENDIX A**

### **Search Syntax 115**

- 116** Why proper search syntax is important
- 117** Phrases, proper names, and capitalization
  - 117** Phrases
  - 117** Proper names
  - 117** Capitalization
- 118** Using plus and minus operators
  - 118** Plus
  - 118** Minus
  - 119** Searches by document title
  - 119** Searching by URL
  - 120** Searching by site
  - 120** Searching by link
  - 120** Searching by image links
  - 120** Searching by "alt" (image)
  - 121** Searching by description or keywords
  - 121** Using "Meta" searches
  - 121** Using Dublin Core "Meta" searches
  - 122** Searching by doctype (document type)
  - 122** Language
  - 122** Searching by collection
  - 123** Searching by topic

	<b>124</b>	The difference between pipe and plus
	<b>125</b>	Double-pipe
<b>INDEX</b>	<b>127</b>	

## Preface



*Products*

This guide describes how to customize your Inktomi Enterprise Search installation.

This preface contains the following sections:

- [Navigating this document](#), on *page 8*
- [Who should read this book](#), on *page 9*
- [Related Reading](#), on *page 10*

## ■ Navigating this document

This document explains how to add Inktomi Search/CCE functions to sites that already have Inktomi Search Software installed. The document contains the following chapters:

- Chapter 1, [Customizing with the Admin Interface](#), describes how you can customize your search installation by changing setting through the admin interface.
- Chapter 2, [Active Doc Directory Customizations](#), describes how to customize your search interface by modifying the appropriate documents in the `/docs` directory.
- Chapter 3, [Customizing with Form Variables](#), describes how you can customize the way in which Inktomi Search Software handles forms by manipulating its form variables.
- Chapter 4, [Integration](#), describes how to integrate frame sets and customize Inktomi Search Software's style sheet.
- Chapter 5, [Changing the patches.py file](#), describes changes you can make to the `patches.py` file.
- Chapter 6, [Using Content Assistants](#), describes how you can create a content assistant to assign topics to indexed documents, filter unwanted or inappropriate documents, or add meta-data to indexed documents.
- Chapter 7, [Customization Examples and Templates](#), provides several examples of customization changes.
- Appendix A, [Search Syntax](#), describes the search syntax.

## ■ Who should read this book

This book is designed for administrators who will be customizing an Inktomi Enterprise Search instance. This document assumes that you are familiar with the basic concepts of Inktomi Search.

## ■ Related Reading

The *Inktomi Enterprise Search 4.0* documentation provides valuable information. You can download it and other documents from:

[http://www.inktomi.com/products/search/support/docs/ultra\\_docs.htm](http://www.inktomi.com/products/search/support/docs/ultra_docs.htm).

## Customizing with the Admin Interface



This chapter explains how to use the Inktomi Search Admin interface to customize your search installation. This chapter contains the following sections:

- [Using the Admin Interface](#), on page **12**
- [Query parameters](#), on page **13**
- [Displaying searchable collections at the query form](#), on page **15**
- [CCE parameters](#), on page **17**
- [CCE initial pages](#), on page **18**
- [CCE topic browse pages](#), on page **18**
- [CCE search pages](#), on page **18**

## ■ Using the Admin Interface

Inktomi Search admin interface provides an easy environment for making simple customizations. Setting values through the admin interface means you do not have to edit any HTML or Python scripts, which reduces the possibility of errors. It is recommended that you use the admin interface for customizations whenever possible.

You can find out more about search administration by reading the *Inktomi Search Software Administrator Guide*, available on the Inktomi web site:

[http://www.inktomi.com/products/search/support/docs/ultra\\_docs.htm](http://www.inktomi.com/products/search/support/docs/ultra_docs.htm)

## ■ Query parameters

Following list describes some simple customizations you can perform through the **Server Parameters** tab of the admin interface, under **Query Parameters**:

- To set the default query mode to “start new search”, “search these results”, or “search the entire Web”, select from the “Default query mode” listbox.
- To set the default value for whether the results pages should hide or show summaries, select from the “Default look” listbox.
- To set the default filter for whether the results should be sorted by date, relevance, or title, select from the “Default results filter” listbox.
- To set the default value for whether the initial search page displays an advanced or a simple search form, select from the “Default query form complexity” listbox.
- To determine whether the query form should be displayed above hits, below hits, or both above and below hits, check the appropriate check boxes for “Show query box above / below the hits”.
- To add or remove values and labels for the listbox items in the Advanced search form, edit the “Advanced query fields” text box. Listings should be in the format `fieldname: , display text`.
- To set the default value for number of hits displayed per page, edit the “Display number of hits” listbox.
- To change the page width, edit the “Results page width” text box. The page width can be set to an integer or a percentage.
- To adjust the number of most relevant query hits to feed into the results filter for processing, edit the “Filter number of hits” text box.
- To set the default filter for whether or not to show word scores, toggle the “Show individual word scores” checkbox.
- To set whether or not Inktomi Search Software should include duplicate URLs as separate hits in the results pages, toggle the “Remove from results duplicate hits with the same URL” checkbox.
- To determine whether Inktomi Search Software should highlight users’ query terms in the title and summary of each hit, toggle the “Highlight query terms in results page” checkbox.
- To set whether or not hits from the same site should appear in clumps, toggle “Interleave Sites” checkbox.
- To determine whether Inktomi Search Software should provide users with the option to search the Internet, in addition to “Start new search” and “Search these results”, toggle the “Provide option to search the Internet” checkbox.

FIGURE 1 Parameters you can set through the admin interface.

### Query Parameters

Default query mode	<div>start new search</div>	Display number of hits	<div>10 hits</div>
Default look	<div>show summaries</div>	Results page width	<div>100%</div>
Default results filter	<div>sort by relevance</div>	Filter number of hits	<div>100</div>
Default query form complexity	<div>simple</div>	Document quality weight	<div>1</div>

Show query form ☒ above ☐ below the hits.

Advanced query fields

, in the body

title:, in the title

url:, in the URL

site:, in the site name

link:, in a link

imagelink:, in an image link

alt:, in image alt text

description:, in the description

☐ Show individual word scores

☐ Remove from results duplicate hits with the same URL

☒ Highlight query terms in results page

☐ Interleave Sites

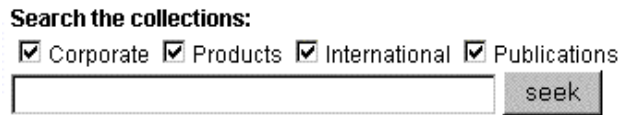
☒ Provide option to search the Internet

## ■ Displaying searchable collections at the query form

In the search interface, collections that are available for search are displayed above the search box.

[Figure 2](#) shows searchable collections.

**FIGURE 2** *Searchable collections at the query form.*



**Search the collections:**  
☒ Corporate ☒ Products ☒ International ☒ Publications

All collections are displayed at the query form by default, but you can manually determine whether Inktomi Search should hide or display, or search or not search, a particular collection. To control which collections are displayed, use the following steps:

- 1 Click on the **Collections/Tuning** tab on the admin interface, shown in [Figure 3](#).
- 2 Choose the collections you wish to modify in the **Collection** listbox, located just below the navigation bar.
- 3 Look under **Spider Tuning Parameters** for the **Show this collection by default** checkbox.
- 4 Un-check the checkbox to hide the current collection; check the checkbox to display the current collection.

The **Show this collection by default** and **Search this collection by default** parameters often work in tandem. However there may be situations where you want to search a particular collection, but not necessarily display that collection to the user.

FIGURE 3 Hide or display collections at the query form.

**Spider Tuning Parameters**

Number of spider threads	<input type="text" value="5"/>	Automatic-merge minimum on-disk index ratio	<input type="text" value="8.0"/>
Maximum document revisit interval (days)	<input type="text" value="32.0"/>	Automatic-merge minimum on-disk index size (documents)	<input type="text" value="5000"/>
Default document age estimate (days)	<input type="text" value="16.0"/>	HTTP connection timeout (seconds)	<input type="text" value="60"/>
Minimum document revisit interval (days)	<input type="text" value="1.0"/>	In-memory index size (documents)	<input type="text" value="200"/>
Number of revisit queues	<input type="text" value="6"/>	Maximum document download (megabytes)	<input type="text" value="1.0"/>
Maximum parse cpu time (seconds)	<input type="text" value="60"/>	Data directory	<input type="text" value="autos"/>
Default Encoding	<input type="text" value="Western (ISO-8859-1)"/>		
Primary Language	<input type="text" value="English"/>	<input checked="" type="checkbox"/> Use HTTP keep-alives	
<input type="checkbox"/> Use sitelist.txt files		<input type="checkbox"/> Use primary host names	
<input type="checkbox"/> Discover sites		<input checked="" type="checkbox"/> Ignore case differences in URLs	
<input checked="" type="checkbox"/> Accept cookies		<input checked="" type="checkbox"/> Remove session IDs from URLs	
<input checked="" type="checkbox"/> Show this collection by default		<input checked="" type="checkbox"/> Search this collection by default	

## ■ CCE parameters

Inktomi Search Software CCE server parameters also allow for easy customization. If your license key enables Inktomi Search Software CCE, you will find these parameters at the bottom of the **Server Parameters** tab on the admin interface, shown in Figure 4. If your license key does not enable Inktomi Search Software CCE, these parameters will not be visible.

To find out more about Inktomi Search Software CCE, visit:

<http://www.inktomi.com/products/search/products/ultraseek/cce/index.html>

FIGURE 4 Inktomi Search Software CCE server parameters.

**CCE Initial Page**

Number of topics columns  Show query form ☒ above ☐ below the topics.

Maximum number of topics children  ☒ Indent topics children.

Show all topics on search home page ☒

**CCE Topic Browse Pages**

Number of subtopics columns  Show subtopics ☒ above ☐ below the hits.

Maximum number of subtopics children  ☒ Indent subtopics children.

Show query form ☒ above ☐ below the hits.

**CCE Search Pages**

Number of related topics columns  Show related topics ☒ above ☐ below the hits.

Maximum number of related topics children  ☒ Indent related topics children.

Maximum number of related topics  ☒ Show full topic path

Group results by topic ☒ ☐ Show inline topic path for each hit.

ok cancel help

## CCE initial pages

The following are some simple customizations that you can perform that affect the look of the Inktomi Search Software CCE initial page:

- Select the number of topics columns
- Select from the maximum number of topics children
- Determine whether the query form should be displayed above topics, below topics, or both above and below topics
- Determine whether topics children should be indented
- Determine whether all topics should be shown

## CCE topic browse pages

CCE topic browse pages are the pages that follow a topic click-through. Following are some simple customizations that you can perform that affect the look of the Inktomi Search Software CCE topic browse pages:

- Select the number of subtopics columns
- Select from the maximum number of subtopics children
- Determine whether the query form should be displayed above or below hits
- Determine whether subtopics should be displayed above or below hits
- Determine whether subtopics children should be indented

## CCE search pages

CCE search pages are those pages that follow a query. Here are some simple customizations that you can perform that affect the look of the Inktomi Search Software CCE search pages:

- Select the number of related topics columns
- Select from the maximum number of related topics children
- Select from the maximum number of related topics
- Determine whether results should be grouped by topic
- Determine whether related topics should be displayed above hits or below hits
- Determine whether related topics children should be indented
- Determine whether the full topic path should be displayed
- Determine whether the inline topic path for each result should be shown

FIGURE 5 Page showing regular topics listing.

Search:

☒ the collections
 

☒ Books
 ☒ Soam's Internal
 ☒ disTest


☐ the Internet

search

[Help](#)
[Advanced](#)

Search by

l n k t o m i



Tip: To search with double .

Example: bas

[Danish](#) - [German](#) - [English](#) - [Spanish](#) - [Finnish](#) - [French](#) - [Italian](#) - [Japanese](#) - [Korean](#) - [Dutch](#) - [Norwegian](#) - [Portuguese](#) - [Swedish](#) - [...](#)

[Expand all topics](#)

### Topics:

#### [Books](#)

[Jane Austen](#)

#### [Docs](#)

[EDB](#)

#### [Search](#)

[cce & ex](#)

FIGURE 6 Page showing topics and sub-topics.

Search:

☒ the collections
 

☒ Books
 ☒ Soam's Internal
 ☒ disTest


☐ the Internet

search

[Help](#)
[Advanced](#)

Search by

l n k t o m i



[Danish](#) - [German](#) - [English](#) - [Spanish](#) - [Finnish](#) - [French](#) - [Italian](#) - [Japanese](#) - [Korean](#) - [Dutch](#) - [Norwegian](#) - [Portuguese](#) - [...](#)

[Show only top level topics](#)

### Topics:

#### [Books](#)

[Jane Austen](#)

[Mansfield Park](#)

#### [Docs](#)

[EDB](#)

[Search](#)

[cce & ex](#)

CHAPTER 1 | CCE parameters | 19

FIGURE 7 Page showing results grouped by topic.

[Start new search](#)
[Search these results](#)
[Search entire Web](#)

Search Books:

search

[Help](#)
[Advanced](#)

Search by

ink

tomi

**Tip:** Separate unrelated proper names with a comma.

**Example:** Bill Gates, Steve Jobs

[Danish](#) - [German](#) - [English](#) - [Spanish](#) - [Finnish](#) - [French](#) - [Italian](#) - [Japanese](#) - [Korean](#) - [Dutch](#) - [Norwegian](#) - [Portuguese](#) - [Swedish](#) - [Simplified Chinese](#) - [Traditional Chinese](#)

**Topic:** [Home](#) > [Books](#)

**Subtopics:**

[Jane Austen](#)

[Mansfield Park](#)

497 results found, sorted by relevance

[score using date](#)
[hide summaries](#)

1-10

**Books**


<p><a href="#">Pride and Prejudice(Eng/Chi)</a></p> <p>Bilingual English/Chinese text of Jane Austen's novel Pride and Prejudice (1813). Texts are side by side for easy comparison and language study.</p> <p><a href="http://nosferatu.inktomi.com:81/jane_austen/pride_and_prejudice_3/pridprej.html">http://nosferatu.inktomi.com:81/jane_austen/pride_and_prejudice_3/pridprej.html</a> - 8.5KB - Books</p>	<p>10 Jul 98</p> <p><a href="#">Find Similar</a></p>
<p><a href="http://nosferatu.inktomi.com:81/jane_austen/PRIDPREJ.TXT">http://nosferatu.inktomi.com:81/jane_austen/PRIDPREJ.TXT</a></p> <p>[This e-text is in the public domain, and has been corrected against the 1923 R.W. Chapman edition, with slight punctuation modernization, by churchh@uts.cc.utexas.edu. Chapman's chronology and ...</p> <p><a href="http://nosferatu.inktomi.com:81/jane_austen/PRIDPREJ.TXT">http://nosferatu.inktomi.com:81/jane_austen/PRIDPREJ.TXT</a> - 707.7KB - Books</p>	<p>09 Mar 96</p> <p><a href="#">Find Similar</a></p>
<p><a href="#">Pride/Prej Chapter 18 (Vol.I, Chap. XVIII)</a></p> <p>Pride and Prejudice by Jane Austen, in English and Chinese translation, Chapter 18 (Vol. I, Chap. XVIII)</p> <p><a href="http://nosferatu.inktomi.com:81/jane_austen/pride_and_prejudice_3/chap0118.html">http://nosferatu.inktomi.com:81/jane_austen/pride_and_prejudice_3/chap0118.html</a> - 69.0KB - Books</p>	<p>10 Jul 98</p> <p><a href="#">Find Similar</a></p>
<p><a href="http://nosferatu.inktomi.com:81/jane_austen/pride_and_prejudice_2/Volume01/chapter18.txt">http://nosferatu.inktomi.com:81/jane_austen/pride_and_prejudice_2/Volume01/chapter18.txt</a></p> <p>Uploaded September 16, 1994 Pride And Prejudice by Jane Austen 1813 &lt;CHAPTER XVIII (18)&gt; TILL Elizabeth entered the drawing-room at Netherfield and looked in vain for Mr. ...</p> <p><a href="http://nosferatu.inktomi.com:81/jane_austen/pride_and_prejudice_2/Volume01/chapter18.txt">http://nosferatu.inktomi.com:81/jane_austen/pride_and_prejudice_2/Volume01/chapter18.txt</a> - 29.3KB - Books</p>	<p>09 Jul 98</p> <p><a href="#">Find Similar</a></p>
<p><a href="#">AUSTEN: Emma 42</a></p> <p>After being long fed with hopes of a speedy visit from Mr. and Mrs. Sucking, the Highbury world were obliged to endure the mortification of hearing that they could not ...</p> <p><a href="http://nosferatu.inktomi.com:81/jane_austen/emma_3/chap42.html">http://nosferatu.inktomi.com:81/jane_austen/emma_3/chap42.html</a> - 28.9KB - Books</p>	<p>09 Jul 98</p> <p><a href="#">Find Similar</a></p>
<p><a href="#">Emma, Chapter42</a></p> <p>After being long fed with hopes of a speedy visit from Mr. and Mrs. Sucking, the Highbury world were obliged to endure the mortification of hearing that they could not ...</p> <p><a href="http://nosferatu.inktomi.com:81/jane_austen/emma_2/chapt42.htm">http://nosferatu.inktomi.com:81/jane_austen/emma_2/chapt42.htm</a> - 29.4KB - Books</p>	<p>09 Jul 98</p> <p><a href="#">Find Similar</a></p>

FIGURE 8 Page showing topic to which each result belongs.

[Start new search](#) Search these results [Search entire Web](#)

**Search EDB:**

  
 [Help](#) [Advanced](#)

Search by 

[Danish](#) - [German](#) - [English](#) - [Spanish](#) - [Finnish](#) - [French](#) - [Italian](#) - [Japanese](#) - [Korean](#) - [Dutch](#) - [Norwegian](#) - [Portuguese](#) - [Swedish](#) - [Simplified Chinese](#) - [Traditional Chinese](#)

**Topic:** [Home](#) > [EDB](#)

12 results found, sorted by relevance [score using date](#) [hide summaries](#) 1-10 ▶

**[EDB Login](#)** 24 Oct 01  
Login to the Editorial DB. Have you registered before? Username: Password: You can login as user guest but you will be unable to insert or delete items in the EDB. Otherwise, register here ... [Find Similar](#)  
**Topic:** [EDB](#)  
**Topic:** [Search](#)  
<http://internal.inktomi.com/~sacharya/EDB/Search/> - 1.0KB - Soam's Internal


**[EDB Schema](#)** 24 Oct 01  
In Oracle, all tables contain a pseudorow called rowid. Doing a "select rowid" on a table will return the rowid of all the rows in the table in string form. Hence, the rowid column is not included in the table ... [Find Similar](#)  
**Topic:** [EDB](#)  
<http://internal.inktomi.com/~sacharya/EDB/EDBSchema.htm> - 6.2KB - Soam's Internal

FIGURE 9 Page showing results without topic path.

[Start new search](#) Search these results [Search entire Web](#)

**Search EDB:**

  
 [Help](#) [Advanced](#)

Search by 

[Danish](#) - [German](#) - [English](#) - [Spanish](#) - [Finnish](#) - [French](#) - [Italian](#) - [Japanese](#) - [Korean](#) - [Dutch](#) - [Norwegian](#) - [Portuguese](#) - [Swedish](#) - [Simplified Chinese](#) - [Traditional Chinese](#)

**Topic:** [Home](#) > [EDB](#)

12 results found, sorted by relevance [score using date](#) [hide summaries](#) 1-10 ▶

**[EDB Login](#)** 24 Oct 01  
Login to the Editorial DB. Have you registered before? Username: Password: You can login as user guest but you will be unable to insert or delete items in the EDB. Otherwise, register here ... [Find Similar](#)  
<http://internal.inktomi.com/~sacharya/EDB/Search/> - 1.0KB - Soam's Internal

**[EDB Schema](#)** 24 Oct 01  
In Oracle, all tables contain a pseudorow called rowid. Doing a "select rowid" on a table will return the rowid of all the rows in the table in string form. Hence, the rowid column is not included in the table ... [Find Similar](#)  
<http://internal.inktomi.com/~sacharya/EDB/EDBSchema.htm> - 6.2KB - Soam's Internal



## Active Doc Directory Customizations



*Products*

You can customize the look and feel of Inktomi Search Software's search interface by modifying the appropriate documents in the /docs directory.

This chapter contains the following information:

- [Understanding the document structure](#), on page **24**
- [Changing headers and footers](#), on page **25**
- [Building initial search pages](#), on page **27**
- [Building search results pages](#), on page **30**
- [Building the help pages](#), on page **43**
- [Getting ready to customize](#), on page **47**
- [Additional search presence](#), on page **49**

## ■ Understanding the document structure

This section contains general information about how Inktomi Search Software builds pages. Carefully read this section and “Getting ready to customize” on page 47, before you begin customizing. These sections include important information about document structure and how to back up your documents before you get started.

Inktomi Search Software builds its user interface by listing several small include files that make up one large HTML page. Each include file is made up of simple HTML and some Python code. The HTML portion of these pages can be easily edited to create a custom user interface that matches your organization’s Web site. The embedded Python code can also be edited to perform various functions within the pages.

Python code uses indentation to separate program blocks, just like C code uses {} (curly braces). When editing any of the Python in the HTML pages, it is **very important** that the original indentation of the file is not corrupted. Altering the indentation can break the Python embedded pages, making them return an error instead of executing.

When examining the content of the include files, you can easily determine which is HTML and which is Python by following this simple rule of thumb: Anything between a `<!--$` and a `-->` is interpreted as Python. There are also Python variables that are referred to within HTML.

In the following HTML example, Python is in bold:

```
<table width="100%"><tr><td><p>
<!--$write(rand.choice(config.tips));-->
</td></tr></table>
```

In the above example, an HTML table is built around a set of random search tips.

Within regular HTML, you may see references to Python variables. These variables are structured as follows:

```
&$variablename;
```

For example:

```
<!--$
pi=3.14159
-->

<b>The value of pi is &$pi;.</b>
```

In the above example, the output would read:

```
The value of pi is 3.14159.
```



Any embedded Python code must be run through a Python interpreter before it can be displayed. Inktomi Search Software includes such an interpreter.

The include files can be found in the active document directory of your Inktomi Search Software installation. By default, /docs is the active document directory. Two basic search page formats are built by Inktomi Search Software:

- Initial search pages
- Search results pages

Take a moment to examine the `/docs` directory. The HTML files contained here make up these two basic search page formats.

### Initial search page

header.html  
coreforma.html  
index.html  
indexform.html  
indexforma.html  
topics.html  
footer.html  
default.css

### Search results pages

header.html  
coreforma.html  
hitsnavtop.html  
query.html  
queryform0.html  
queryform0a.html  
queryform1.html  
queryform1a.html  
queryform2.html  
queryform2a.html  
topics.html  
relatedtopics.html  
subtopics.html  
hits.html  
onehit1.html  
onehit2.html  
findsimilar.html  
hitsnavbottom.html  
nohits.html  
thesaurus.html  
make\_ad.html  
footer.html  
default.css

Also found within the `/docs` directory are the `/help` and `/images` subdirectories. The included files that make up Inktomi Search Software's User help section can be found in the `/help` subdirectory. These files are explained in [Building the help pages](#), on page 43. The images that make up Inktomi Search Software's search interface can be found within the `/images` subdirectory.

Notice that the files `header.html` and `footer.html` help to build both the initial search page and the search results page.

## Changing headers and footers

The basic structure of an Inktomi Search Software user page includes a header, a footer, and a section for queries. Although the query page itself is quite extensive, the header and footer sections are very simple. To significantly change the overall look of the page, change the files for the header and footer: `header.html` and `footer.html`.

For example, if you want to add your company's logo to the top of all search pages, add the appropriate HTML to your `header.html` file, and the logo will appear at the top of all initial search pages and on all the search results pages. Content added to `footer.html` will be displayed at the *bottom* of all these pages.

To save resources, Inktomi Search Software caches its interface documents. When modifying any of Inktomi Search Software's HTML files, it is important to reload the pages using the admin interface so

that your changes will take effect. It is not necessary to restart the server. To reload the pages through the Web server, go to **Server Parameters** in the admin interface and click **RELOAD**.

## Tables

The look of the search results has been designed so that users with browsers that do not support tables can still easily scan search results. However, you must use a browser that supports tables in the admin interface.

Much of the interface for both the initial search pages and the search results pages is made up of nested tables. When customizing, it is important to take note of the table structure.

The outermost table is composed of two cells, one comprises 67% of the table width, the other 33%. The exact width of the table will vary according to the “Results page width” parameter setting under **Server Parameters** of the admin interface. The width specified here will be sent as a Python variable to the “width” attributes of corresponding “table” tags. See [Figure](#) , on [page 14](#) for more information.

The proportion of the table cells can be modified to fit your interface by simply modifying the HTML. A common modification is to put the rotating search tips on a separate row, allowing for another table cell on either side of the search box.

## ■ Building initial search pages

Inktomi Search Software uses `index.html` to build all variations of the initial search page. When you look at the `index.html` source, you will see Python calls that include certain files. The Python calls look something like this:

```
exec.self.file("header.html")
```

In the above example, `header.html` is called. Following are the include files `index.html` calls:

- `header.html`
- one file from the `indexform_.html` series
- `footer.html`
- `topics.html` (if using CCE)

### The `indexform_.html` series

There are only two files in this series: `indexform.html`, and `indexforma.html`. Inktomi Search Software chooses which file to use based on the following method:

- Inktomi Search Software uses `indexform.html` to create a Simple Search form. In a Simple Search form, the user simply selects collections to search and types a query into the search box. Rotating tips are available to assist the user in creating proper search syntax.

FIGURE 10 Initial search page built with `indexform.html`.

Search: ☒ the collections  
☒ Car reviews ☒ Inktomi Search documentation ☒ Inktomi Search FAQs  
☐ the Internet

[Help](#) [Advanced](#)

Search by Inktomi

**Tip:** You may use accented European characters for a more exact match.

**Examples:** café, piñata

- Inktomi Search Software uses `indexforma.html` to create an Advanced Search form. In an Advanced Search form, the user has tools that assist him in performing queries that have excellent search syntax. There are no rotating tips in the Advanced Search form, as they are unnecessary. If you have a license key that enables multiple languages, the Advanced Search form will also allow the user to select a language.

FIGURE 11 Initial search page built with `indexforma.html`.

The screenshot shows the Inktomi Search Advanced Search form. It includes a 'Search:' section with radio buttons for 'the collections' (selected) and 'the Internet'. Under 'the collections', there are three checked checkboxes: 'Car reviews', 'Inktomi Search documentation', and 'Inktomi Search FAQs'. Below this, there are three sections for document filtering: 'for documents that', 'and', and 'and'. Each section has a dropdown menu for the filter type (e.g., 'should contain', 'must contain', 'must not contain'), a dropdown for the location (e.g., 'in the body'), and a dropdown for the words (e.g., 'the words'). There are also text input fields for each section. The 'dated' section has radio buttons for 'Anytime' (selected), 'in the last week', 'on or after' (with date fields for 7 March 2001), and 'and before' (with date fields for 14 March 2001). The 'and show' section has dropdowns for '10 results', 'sorted by relevance', and 'with summaries'. There is a checkbox for 'Show individual word scores'. At the bottom left is a 'search' button and links for 'Help' and 'Simple'. At the bottom right is the 'Search by Inktomi' logo.

## Topics

If Inktomi Search Software CCE is enabled through the license key, Inktomi Search Software then calls `topics.html`. The file `topics.html` lists topics built by Inktomi Search Software CCE. Whether Inktomi Search Software lists topics before or after the query form depends on the settings you designate in the CCE parameters section of **Server/Parameters** on the admin interface. Figure 12 shows an example of a customized CCE initial page.

FIGURE 12 Initial search page showing topics.

Search: ☒ Musical Instruments


☐ the Internet

search

[Help](#) [Advanced](#)

Search by

inktom



*Tip: You can type in your query using plain language or just use keywords.*

**Example:** who makes the best wine?

[Danish](#) - [German](#) - [English](#) - [Spanish](#) - [Finnish](#) - [French](#) - [Italian](#) - [Japanese](#) - [Korean](#) - [Dutch](#) - [Norwegian](#) - [Portuguese](#) - [Swedish](#) - [Simplified Chinese](#) - [Traditional Chinese](#)

Topics:

[Electronic Instruments](#)

[Drum Machines](#), [Samplers](#), [Synthesizers](#), ...

[Keyboards](#)

[Harpsichord](#), [Organ](#), [Piano](#), ...

[Percussion](#)

[Drum Kits](#), [Drum Machines@](#), [Timpani](#)

[String Instruments](#)

[Bass](#), [Guitar](#), [Mandolin](#), ...

[Wind Instruments](#)

[Brass Instruments](#), [Double Reed Instruments](#), [Woodwind Instruments](#)

## ■ Building search results pages

Search results pages are generated through an embedded Python script that calls the search engine and gets the search results in the form of a list. The embedded script sorts the results list before displaying the results. Each element in the results list includes the following:

- Title
- Summary
- URL
- Publisher
- Date
- Size

Just as Inktomi Search Software uses `index.html` to build the initial search pages, it uses `query.html` to build search results pages. The file `query.html` tells Inktomi Search Software to construct a page that uses the following include files:

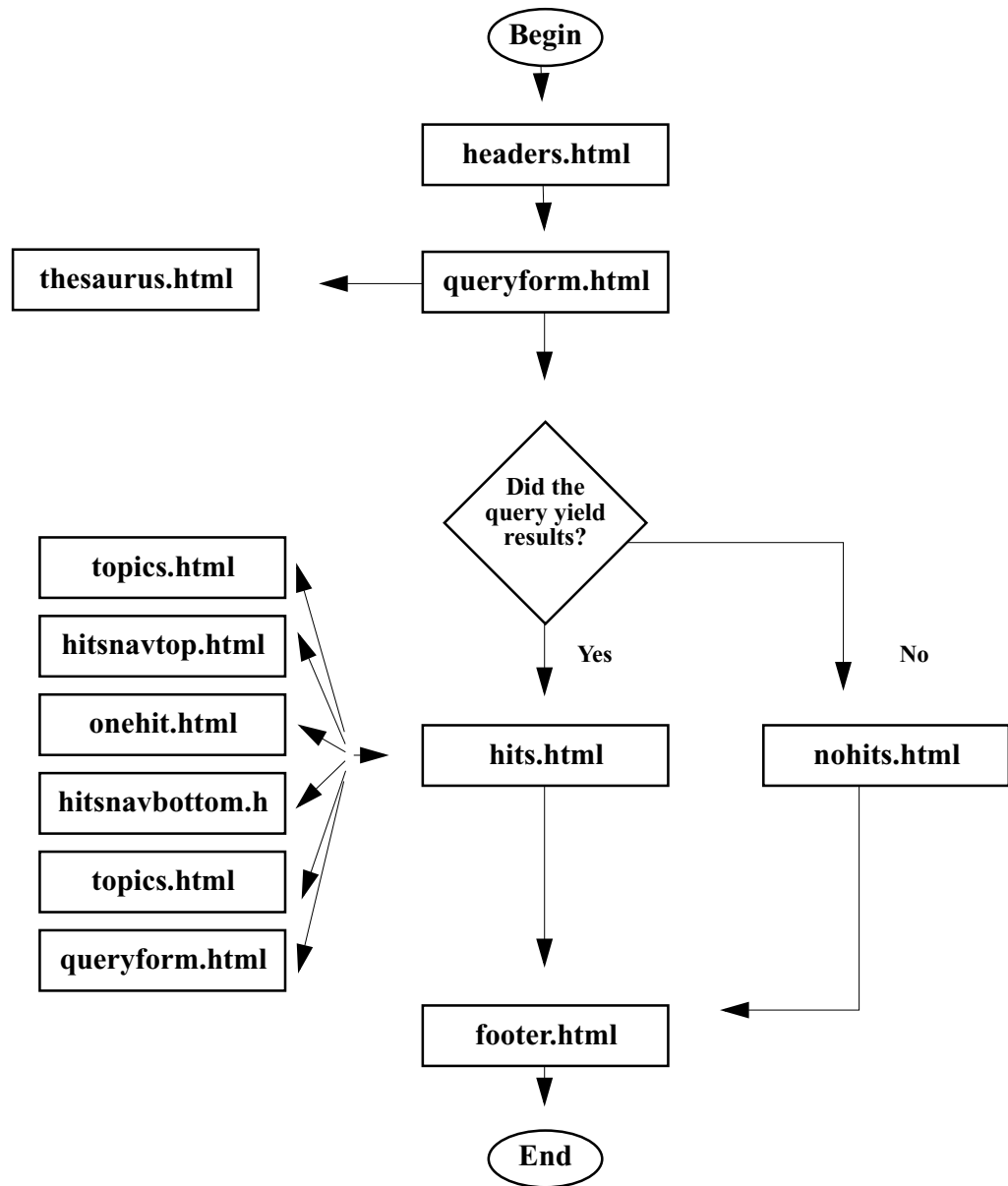
- `header.html`
- one file from the `queryform_`.html series
- (if a thesaurus match is found) `thesaurus.html`
- (if Inktomi Search Software CCE is enabled) `topics.html` and `relatedtopics.html` or `subtopics.html`
- either `hits.html` or `nohits.html`
- `hitsnavtop.html`
- either `onehit1.html` or `onehit2.html`
- `hitsnavbottom.html`
- `footer.html`

The file `query.html` first tells Inktomi Search Software to display the file `header.html` and (if settings permit) one of the files from the `queryform_`.html series (the search box). If Inktomi Search Software CCE is enabled, Inktomi Search Software then displays `relatedtopics.html` or `subtopics.html`, if settings permit.

If no search results are found, `nohits.html` is displayed next, followed by `footer.html`.

If search results are found, `hits.html` is called along with its associated files. Then, if settings permit, Inktomi Search Software again displays one of the files from the `queryform_`.html series (the search box) and then displays `relatedtopics.html` or `subtopics.html`, if settings permit. Finally, Inktomi Search Software displays `footer.html`, shown in [Figure 13](#).

FIGURE 13 How `query.html` builds search results pages.



## The `queryform_.html` series

In the `queryform_.html` series, there are six files:

- `queryform0.html`
- `queryform0a.html`
- `queryform1.html`
- `queryform1a.html`

- `queryform2.html`
- `queryform2a.html`

Modify the `queryform_`.html series to alter the following:

- search box
- rotating tips
- “powered by Inktomi Search Software” logo
- “Start new search”
- “Search these results”
- “Search entire Web”

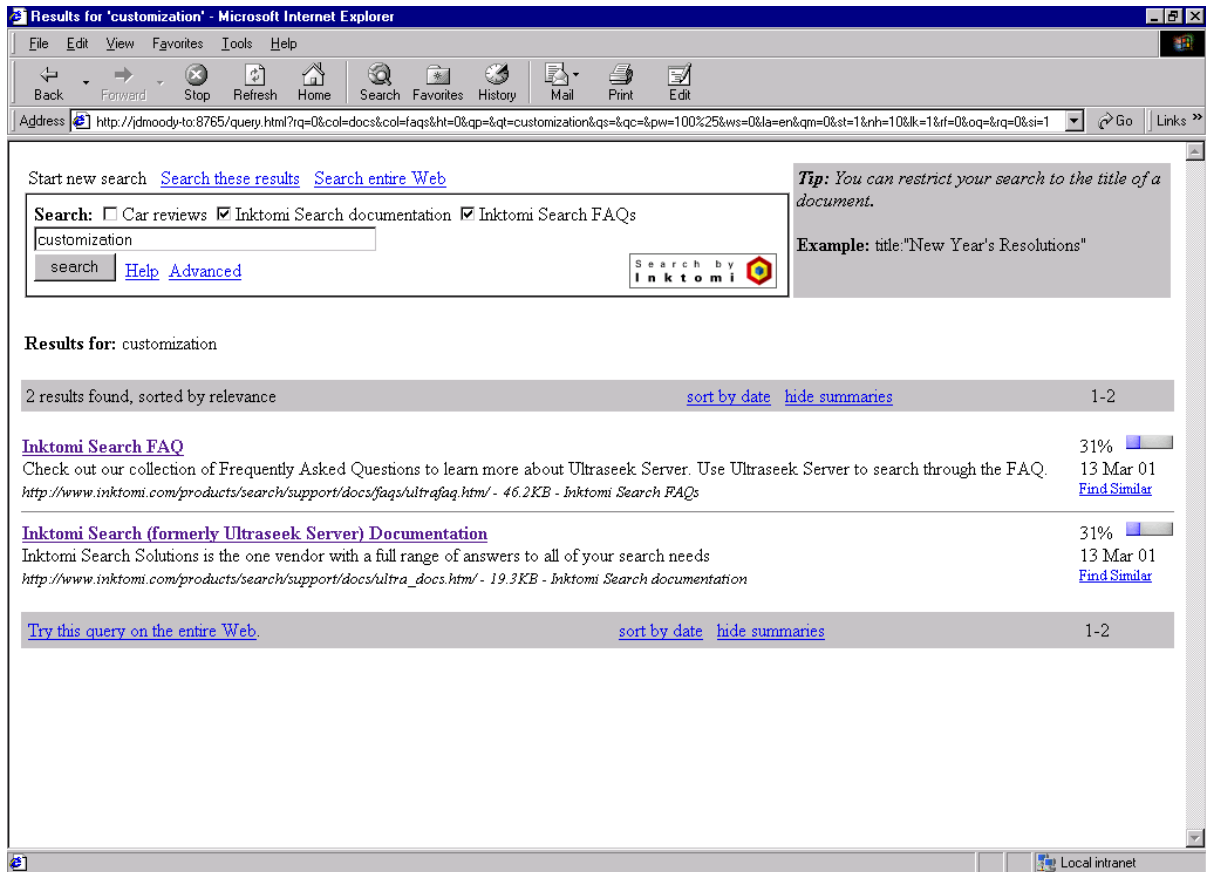
Inktomi Search Software may call a file from the `queryform_`.html series at various places: it may call it before `hitsnavtop.html` or after `hitsnavbottom.html`. Whether and where these files are called depends on the settings you designate in the query parameters section of Server Parameters in the admin interface.

The differences between the files in the `queryform_`.html series lie in whether the query form is Simple or Advanced, and which of the three search options is selected (“Start new search”, “Search these results”, or “Search entire Web”). Inktomi Search Software will use one of several forms depending on the circumstances.

The following circumstances help determine which form Inktomi Search Software will use:

- The file `queryform0.html`, shown in [Figure 14](#), returns an interface that uses the Simple Search form and has the Start New Search option selected. All subsequent queries performed by the user will be independent of previous queries.

FIGURE 14 Search results page built with queryForm0.html.



- The file queryform0a.html, shown in Figure 15, returns an interface similar to queryform0.html, except that the Advanced Search form is used. The Start New Search option is selected, so all subsequent queries performed by the user will be independent of previous queries.

FIGURE 15 Search results page built with queryform0a.html.

Search: ☐ Car reviews ☒ Inktomi Search documentation ☒ Inktomi Search FAQs

for documents that

should contain

in the body

the words

and

must contain

in the body

the words

and

must not contain

in the body

the words

dated

☒ Anytime

☐ in the last week

☐ on or after 

7

March

2001

and before 

14

March

2001

and show

10 results

sorted by relevance

with summaries

☐ Show individual word scores

search

[Help](#) [Simple](#)

Search by

Inktomi

Results for: customization

2 results found, sorted by relevance		<a href="#">sort by date</a> <a href="#">hide summaries</a>	1-2
<a href="#">Inktomi Search FAQ</a>			31% <div></div>
Check out our collection of Frequently Asked Questions to learn more about Ultraseek Server. Use Ultraseek Server to search through the FAQ.			13 Mar 01

- The file `queryform1.html` returns an interface that uses the Simple Search form and has the **Search These Results** option selected. All subsequent queries performed by the user will be a subset of the previous query.

FIGURE 16 Search results page built with `queryform1.html`.

[Start new search](#)
[Search these results](#)
[Search entire Web](#)

**Search the results of:** customization

for documents that

should contain in the body the words

and

must contain in the body the words

and

must not contain in the body the words

dated

☒ Anytime

☐ in the last week

☐ on or after 7 March 2001


and before 14 March 2001

and show

10 results sorted by relevance with summaries


☐ Show individual word scores

[Help](#) [Simple](#)

Search by 

**Results for:** customization

2 results found, sorted by relevance [sort by date](#) [hide summaries](#) 1-2

[Inktomi Search FAQ](#) 31% 

- The file `queryform1a.html` returns an interface similar to `queryform1.html`, except that the Advanced Search form is used. The **Search These Results** option is selected, so all subsequent queries performed by the user will be a subset of the previous query.

FIGURE 17 Search results page built with `queryform1a.html`.

[Start new search](#) [Search these results](#) [Search entire Web](#)

Search the results of: customization

search

[Help](#) [Advanced](#)

Search by  
Inktomi

*Tip: You can use + in front of a term to require it.*

**Example:** + "scuba diving", Hawaii, Maui

Results for: customization

2 results found, sorted by relevance

[sort by date](#) [hide summaries](#)

1-2

[Inktomi Search FAQ](#)

31%

Check out our collection of Frequently Asked Questions to learn more about Ultraseek Server. Use Ultraseek Server to search through the FAQ.  
<http://www.inktomi.com/products/search/support/docs/fags/ultrafaq.html/> - 46.2KB - Inktomi Search FAQs

13 Mar 01

[Find Similar](#)

[Inktomi Search \(formerly Ultraseek Server\) Documentation](#)

31%

Inktomi Search Solutions is the one vendor with a full range of answers to all of your search needs  
[http://www.inktomi.com/products/search/support/docs/ultra\\_docs.html/](http://www.inktomi.com/products/search/support/docs/ultra_docs.html/) - 19.3KB - Inktomi Search documentation

13 Mar 01

[Find Similar](#)

[Try this query on the entire Web.](#)

[sort by date](#) [hide summaries](#)

1-2

- The file `queryform2.html` returns an interface that uses the Simple Search form and has the **Search Entire Web** option selected. All subsequent queries performed by the user will search the entire World Wide Web.

FIGURE 18 Search results page built with `queryform2.html`.

[Start new search](#) [Search these results](#) Search entire Web


Search entire Web:

search

[Help](#) [Advanced](#)

Search by

Inktomi



**Tip:** Separate unrelated proper names with a comma.

**Example:** Bill Gates, Steve Jobs

Results for: customization

2 results found, sorted by relevance

[sort by date](#) [hide summaries](#)

1-2

[Inktomi Search FAQ](#)

Check out our collection of Frequently Asked Questions to learn more about Ultraseek Server. Use Ultraseek Server to search through the FAQ.  
<http://www.inktom.com/products/search/support/docs/faqs/ultrafaq.htm/> - 46.2KB - Inktomi Search FAQs

31%

13 Mar 01

[Find Similar](#)

[Inktomi Search \(formerly Ultraseek Server\) Documentation](#)

Inktomi Search Solutions is the one vendor with a full range of answers to all of your search needs  
[http://www.inktom.com/products/search/support/docs/ultra\\_docs.htm/](http://www.inktom.com/products/search/support/docs/ultra_docs.htm/) - 19.3KB - Inktomi Search documentation

31%

13 Mar 01

[Find Similar](#)

[Try this query on the entire Web.](#)

[sort by date](#) [hide summaries](#)

1-2

- The file `queryform2a.html` returns an interface similar to `queryform2.html`, except that the Advanced Search form is used. The **Search Entire Web** option is selected, so all subsequent queries performed by the user will search the entire World Wide Web.

FIGURE 19 Search results page built with `queryform2a.html`.

[Start new search](#) [Search these results](#) [Search entire Web](#)

Search entire Web for documents that

should contain

in the body

the words

and

must contain

in the body

the words

and

must not contain

in the body

the words

dated

☒ Anytime

☐ in the last week

☐ on or after 7 March 2001

and before 14 March 2001

and show

10 results

sorted by relevance

with summaries

search

[Help](#) [Simple](#)

Search by  
i n k t o m i

Results for: customization

2 results found, sorted by relevance

[sort by date](#) [hide summaries](#)

1-2

[Inktomi Search FAQ](#)31%13 Mar 01

Check out our collection of Frequently Asked Questions to learn more about Ultraseek Server. Use Ultraseek Server to search through the FAQ.

## Thesaurus

If a user's query contains terms that Inktomi Search Software finds in its thesaurus, Inktomi Search Software calls `thesaurus.html`. The file `thesaurus.html` displays alternative search terms, and the user has the option of either expanding or not expanding his query based on those terms. On such occasions, Inktomi Search Software displays checkboxes that allow users to do one of the following:


- Individually select terms to expand their search
- Select all query expansion terms at once
- Not select any query expansion terms at all

FIGURE 20 Search interface with thesaurus.html employed.

Start new search [Search these results](#) [Search entire Web](#)

**Search:**

**Click to expand search:**  
☐ bilberry ☐ whortleberry  
 [Help](#) [Advanced](#)

Search by  
i n k t o m i 

[Danish](#) - [German](#) - [English](#) - [Spanish](#) - [Finnish](#) - [French](#) - [Italian](#) - [Japanese](#) - [Korean](#) - [Dutch](#) - [Norwegian](#) - [Portuguese](#) - [Swedish](#) - [Simplified Chinese](#) - [Traditional Chinese](#)

**Results for:** blueberry

**No results were found for your search.**  
Try changing some of the words in your query.

**Tips for better results:**

- Check your [syntax](#).
- Make sure to use the correct spelling.
- Use the correct case (uppercase for capitalized names, lowercase for common words and phrases).  
For example: **Bill Clinton, Boris Yeltsin**
- Try synonyms and variations of words.  
For example: **CDROM, CD-ROM**

**Tip:** To search for a phrase, surround the words with double quote marks.

**Example:** baseball "Hall of Fame"

## Hits or no hits

If the query performed by the user yields results, query.html displays information such as

- query term
- document count
- topics
- word scores

and then tells Inktomi Search Software to call hits.html. The file hits.html in turn calls files that make up the search results. The hits.html file tells Inktomi Search Software to display hitsnavtop.html, relatedtopics.html or subtopics.html if settings allow, several iterations of onehit1.html or onehit2.html, and hitsnavbottom.html.

The file hitsnavtop.html is the file in which contains the top navigational search elements. These elements are located in a gray box, and include such items as:

- Number of results found
- Sorting status
- Sort options
- Page navigation

Modify this file to change the **Sort by Date** and **Hide Summaries** links, or modify font properties for text within and above the gray box.

FIGURE 21 Navigational elements generated by `hitsnavtop.html`.

3 results found, sorted by relevance [sort by date](#) [hide summaries](#) 1-3

If Inktomi Search Software CCE is enabled through the license key, Inktomi Search Software may then call either `relatedtopics.html` or `subtopics.html`. Inktomi Search Software calls `relatedtopics.html` when the search results are in a topic search mode. Inktomi Search Software calls `subtopics.html` when the search results are in a browse or topic search mode. The mode reflected by the search results will depend on whether the user is performing queries or browsing. Inktomi Search Software may call `relatedtopics.html` or `subtopics.html` at various places: it may call it before `hitsnavtop.html` or after `hitsnavbottom.html`. Whether and where these files are called depends on the settings you designate in the CCE parameters section of **Server/Parameters** in the admin interface.

The `onehit` file is spawned repeatedly, once for each hit. The number of times the `onehit` file is called per page depends on how many search results are displayed per page. The `onehit` file may be `onehit1.html`, or `onehit2.html`. Inktomi Search Software calls `onehit1.html` if the setting is on “Show summaries”. Inktomi Search Software calls `onehit2.html` if the setting is on “Hide summaries”. Edit the `onehit` file to change elements of individual search results, such as:

- font properties
- Find Similar
- document date
- document URL
- relevance ranking

The file `hitsnavbottom.html` contains navigational elements similar to `hitsnavtop.html`. The `hitsnavbottom.html` contents are displayed in a colored (usually gray) box at the bottom of the page. Edit this file to change **Sort by Date** and **Hide Summaries** links and font properties within the colored box.

FIGURE 22 Navigational elements generated by `hitsnavbottom.html`.


[Try this query on the entire Web.](#) [sort by date](#) [hide summaries](#) 1-3

If the query performed by the user does not yield results, Inktomi Search Software calls `nohits.html` instead of `hits.html`. The file `nohits.html` consists mostly of HTML, and is very easy to edit. This file explains to the user that no results were found for their search, and gives suggestions on how to improve the query syntax.

FIGURE 23 No Results Found page generated by nohits.html.

Start new search [Search these results](#) [Search entire Web](#)

<b>Search:</b> <input checked="" type="checkbox"/> Car reviews <input checked="" type="checkbox"/> Inktomi Search documentation <input checked="" type="checkbox"/> Inktomi Search FAQs
<input type="text" value="CompuServe"/>
<input type="button" value="search"/> <a href="#">Help</a> <a href="#">Advanced</a>

Search by  
**inktom i** 

**Tip:** Search terms are stemmed.

**Example:** network matches networks and networked.

Results for: CompuServe

No results were found for your search.  
Try changing some of the words in your query.

**Tips for better results:**

- Check your [syntax](#).
- Make sure to use the correct spelling.
- Use the correct case (uppercase for capitalized names, lowercase for common words and phrases).
- Use a comma to separate capitalized names or phrases.  
For example: **Bill Clinton, Boris Yeltsin**
- Try synonyms and variations of words.  
For example: **CDROM, CD-ROM**

Many Inktomi Search Software administrators edit the nohits.html file to include tips and examples that are specific to their organization.

For example, a company called Widget Corporation might replace this:

Try synonyms and variations of words.  
For example: **CDROM, CD-ROM**

with this:

Try synonyms and variations of words.  
For example: **widget gadget thingamajig**

## Search results page builder quick reference

Figure 24 shows the various sections that are built in a search results page. When you want to modify a certain element of the search results, you can use this as a quick reference guide that tells you what file to modify for each section of the search results.

FIGURE 24 Search results page builder quick reference.

The screenshot shows the Inktomi search results page for the query 'inktomi'. The page layout includes a top navigation bar, a left sidebar with links like 'Solutions', 'Partners', and 'Download', and a main content area. Numbered annotations on the right side of the page point to specific elements:

- 1** points to the top navigation bar containing links: HOME, ABOUT INKTOMI, NEWS & EVENTS, PRODUCTS.
- 2** points to the search input field containing the text 'inktomi'.
- 3** points to the 'Click to expand search:' section, which includes checkboxes for 'Inktomi Search' and 'Inktomi', a 'search' button, and links for 'Help' and 'Advanced'.
- 4** points to the 'Results for: inktomi' section, which displays '994 results found, top 500 sorted by relevance' and includes sorting options like 'sort by date' and 'hide summaries'.
- 5** points to the list of search results, which includes entries for 'Inktomi Corporation', 'Inktomi Search Solutions', 'Inktomi Webcasts', and 'Inktomi - Network Products', each with a brief description, a date, and a 'Find Similar' link.

- 1 header.html - This is blank by default, but Figure 24 illustrates a customized header.
- 2 One file from the queryform\_.html series. queryform0.html is shown in Figure 24.
- 3 thesaurus.html
- 4 hitsnavtop.html
- 5 onehit file. onehit1.html is shown in Figure 24.

## ■ Building the help pages

Inktomi Search Software's help pages are located within the `/docs` directory under `/help`.

### Help header and footer files

The `help` directory contains header and footer files, which behave much like their sister files in the `/docs` directory. Similarly, they can be used to dramatically change the interface of all help pages.

The header and footer files of the help interface are slightly different from those of the search interface. In the search interface, there is only one of each of the following files:

- `header.html`
- `footer.html`

In the help interface, there are three of each of the following files:

- `header.html`
- `header1.html`
- `header2.html`
- `footer.html`
- `footer1.html`
- `footer2.html`

In the search interface, header and footer files contain the opening and closing `HTML`, `HEAD`, `TITLE`, and `BODY` container tags. In the help interface, the header and footer files also contain the table structure for the entire help interface, navigational elements, and graphic pointers.

The files `header1.html`, `header2.html`, `footer1.html` and `footer2.html` contain a navigation bar for the help interface. The files `header1.html` points to a smaller graphic, while `header2.html` points to a standard-size graphic. The files `footer1.html` and `footer2.html` contain closing `HTML` tags, with no graphic pointers or table structure.

It is in the `header.html` file and the `footer.html` file that you will find the bulk of the header and footer content. When modifying the header and footer files, be especially mindful of the table structure. Changing elements of the `TABLE` tags can greatly change the entire look of the help interface. Tables and nested tables are stretched across the header, footer, and content files.

Most images in the help interface are listed in the header files. The background image, the vertical gray fade-in bar running down the left side, is specified in the `BODY` tag. The toggle buttons that switch from the help interface to the search interface are specified in the first enclosed table in `header.html`. Other graphic elements, such as the linear graphics and active page indication arrows, are found throughout the help files.

In the help interface, there is a column running down the right side that contains navigational links. These links are generated by Python code found in a nested table in `footer.html`, shown in [Figure 25](#). This code tells Inktomi Search Software to list the active page in plain text with an arrow image used as an indicator, and to list inactive pages as hyperlinks. It also designates which links are allowed for search under the "Security through address-based level specifications" designation.

FIGURE 25 footer.html Python code.

```
<!--$
links = []
links.append(('../help/', 'Quick Tips and Examples', ALLOW_SEARCH))
links.append(('refine.html', 'Refining a Search', ALLOW_SEARCH))
links.append(('special.html', 'Special Searches', ALLOW_SEARCH))
links.append(('boolean.html', 'Requiring or Excluding
Terms', ALLOW_SEARCH))
links.append(('meta.html', 'Meta Tags', ALLOW_SEARCH))
links.append(('syntax.html', 'Search Syntax Summary', ALLOW_SEARCH))
links.append(('copyright.html', 'About Ultraseek Server', ALLOW_SEARCH))
for lnk,txt,aclvl in links:
    if self.access_control_level<aclvl: continue
    --><tr><td><img alt=""
<!--$
        if txt == title:
            -->src="/images/arrow.gif" align=top width=6 height=10
border=0></td>
<!--$
            --><td><font color="#551a8b">&$txt;</font></td></tr>

<!--$
        else:
            -->src="/images/dot_clear.gif" align=top width=6 height=10
border=0></td>
<!--$
            --><td><a href="&$lnk;">&$txt;</a></td></tr>
```

## Help content files

In addition to header.html and footer.html, there are several other files that serve help content:

- searchtips.html
- refine.html
- special.html
- boolean.html
- meta.html
- syntax.html
- copyright.html

At the beginning of each file, there is a bit of Python code that tells Inktomi Search Software to first display header.html, then display the content of the current file, and finally, display footer.html.

copyright.html contains a few simple Python references, but is mostly straightforward HTML. The remaining files contain Python calls to a set of language-specific, content include files, or to default English versions, if no other language versions are present.

```
try: file = self.file('[filetype]_'+la+'.html')
except IOError: file = self.file('[filetype]_en.html')
except OSError: file = self.file('[filetype]_en.html')
exec file
```

The `[filetype]` is generic, and is replaced in each file with the respective content include filename, and the `+la+` refers to the `la` search variable. See [Advanced Search variables](#), on page **71** for more information on the `la` variable).

For example, `'searchtips_'+la+'.html'` would be in the `searchtips.html` Python code.

This coding allows these pages to be language and content customized. Each of the above files (except `copyright.html`) has a corresponding English version, by default.

- `searchtips_en.html`
- `refine_en.html`
- `special_en.html`
- `boolean_en.html`
- `meta_en.html`
- `syntax_en.html`

The help files are available in multiple languages and come with the additional language packs. Please contact your sales representative for more information about adding multiple languages to Inktomi Search.

Aside from the snippets of Python that expand `header.html` and `footer.html`, all of the above files contain simple HTML and are very easy to modify. A very common modification of these content files is to change the help examples so they resemble the rest of the site.

For example, in `searchtips.html`, the Widget Corporation may choose to change this text:

Search by typing words and phrases.  
Pentium computer with 8x CD-ROM for sale

to this:

Search by typing words and phrases.  
Deluxe edition widget with green and white fold-out wings

By displaying search examples that resemble the site's content, the help files are better integrated into the site.

## Other files in the `/help` directory

In addition to the previously mentioned files, the `/help` directory also contains the following files:

- `addurl.html`
- `addurlgo.html`
- `revisitsite.html`
- `revisitsitego.html`
- `revisitsitehelp.html`
- `sites.html`
- `sitesgo.html`
- `urls.html`
- `urlstatus.html`
- `urlstatusgo.html`

## ■ index.html

Each of these files performs a separate function. The files `addurl.html`, `addurlgo.html`, `urlstatus.html`, and `urlstatusgo.html` have functions that serve both administrators and users. Sometimes it is not appropriate to allow users access to the add URL and URL status interfaces. In these cases, modify the corresponding links so that they are only visible to administrators. See [Hiding links under the More Services heading](#), on page 46. The interface for these files can be modified through `header.html` and `footer.html`. However, further modifications to these files is not advised.

Submission of new URLs to the index are allowed by `addurl.html` and `addurlgo.html`. URLs are accepted by `addurl.html`, and `addurlgo.html` confirms the submission and states whether or not the submission is allowed by the filter.

Each file in the `revisitsite` series manages one aspect of a site revisit. The “Revisit site” link takes you to `revisitsite.html`, which allows you to revisit a site. The form submission is then sent to `revisitsitego.html`. Clicking the Help link will take you to `revisitesitehelp.html`.

The “View Sites” link takes you to `sites.html`, where you get a list of all collections Inktomi Search Software has in its index. You will also see the indexing status of each collection, and how many sites are included. Click one of the hyperlinked collections to get to `sitesgo.html`, which includes a list of all sites in the collection and their corresponding status.

The files `urlstatus.html` and `urlstatusgo.html` allow users and administrators to check the indexing status of a given URL. The form is submitted in `urlstatus.html`, and results are provided in `urlstatusgo.html`.

The file `index.html` is the default page for the help interface. The help links throughout the search and admin interfaces point to this page. The `index.html` file is a very small file that simply tells Inktomi Search Software to execute `searchtips.html`, which in turn also executes `header.html` and `footer.html`.

## Hiding links under the More Services heading

The More Services heading is found part-way down the navigational link column running down the right side of the help interface. These links point to semi-administrative pages which may or may not be suitable for all users to navigate. The Server Administration link points to the admin interface, which is suitable only for administrators to navigate.

Sometimes an organization decides it is appropriate to hide links found under the More Services heading. With the links hidden, access to the corresponding areas is further restricted.

Any IP address that is restricted in the admin interface to **Add URL** will not see the “Server Administration” links. If the IP address is restricted to **Search Only**, it will also not see the “Add URL” or “URL Status” links. Using these IP address restrictions is simpler than customizing the Help pages to remove the links.

If the IP address of administrators’ client machines is known, hiding these links can easily be accomplished on the Server/Users tab of the admin interface under “Address-based access level specification”.

## ■ Getting ready to customize

Before customizing Inktomi Search Software's interface, it is important to create back-ups of the existing `/docs` directory. Creating a back-up will save time and provide for easy restoration of defunct files, if necessary. In this section, two new copies of the `/docs` directory are created.

### Protecting customized files during upgrades

Generally, you can use the same customized files from point to point within a version. For example, you can use the same customized files for both version 3.0.x and 3.1.x of Inktomi Search Software. However, since document structure tends to change significantly with new version releases, you generally need to redo your customizations from version to version. If you have done extensive customizations in 3.x.x, you probably need to redo your customizations for version 4.x.x. You'll also need to update your modifications from version 4.0.x to version 4.1.x, since version 4.1.x uses Python version 2.0. (Earlier Inktomi Search Software versions use Python 1.6.)

During an upgrade, it is important to back up customized files. We suggest adding a `.old` to the filenames of your customized files. These can be used for later reference.

### Creating a new active document directory

The first step is to create a directory that has contents that are similar to your `/docs` directory. (For these examples, assume we are working on a Unix-based system, in the `/opt` directory.) The new directory should be located outside the Inktomi Search Software directory structure, so that your files are protected during upgrades and un-installs. The new directory you create will become the active document directory. The active document directory is the directory that Inktomi Search Software uses to create the user interface. By default, the active document directory is the `/docs` sub-directory in the Inktomi Search Software installation directory.

Create a new directory called `/custom`. The new `custom` directory should be located outside the Inktomi Search Software directory structure. **Copy the entire contents of the `/docs` directory to the `/custom` directory.** Once `/custom` becomes the new active document directory, Inktomi Search Software will look to `/custom` to serve the files that create the user interface. If it does not find the files it needs, Inktomi Search Software will get the necessary files from its compiled files.

Create a sub-directory of `/custom` and name it `/test`. Copy the contents of `custom` into `/test`. At this point, there are three possible document directories: one old directory named `/docs`, a new directory named `/custom`, and a new subdirectory of `custom` named `/test`.

### Accessing content in the new directories

To access the new content, you change the path for the active document directory. To change this path, you go to Server Administration. Define the new path in the text box labeled "Document directory". In the above example, the new directory path is `/opt/custom/`.

It is now possible to access both the `custom` and `test` interfaces; at this point, they are still identical. Files in the `/custom` directory can be found at your normal search URL:

[http://\[your search server hostname and port\]/](http://[your search server hostname and port]/)

Files in the `test` directory can be found one directory deeper:

[http://\[your search server hostname and port\]/test/](http://[your search server hostname and port]/test/)

Users will now see the `/custom` directory.

## Editing for customization

Open a browser window, and point it at the **Server/Parameters** tab in the admin interface. Now, open a new browser and point it at the `/test` directory, using the address above. You can now make changes to files in `test` by opening them in an HTML editor. Be sure to use an editor that will not create syntax that conflicts with Inktomi Search Software's pages. (Avoid Microsoft's FrontPage or Netscape Composer for editing these particular pages.) Once you have made and saved your changes, click the **RELOAD** button next to the "Document directory" heading. To view changes, click **RELOAD** or **REFRESH** on the browser toolbar where the `/test` directory document is displayed.

## Publishing changes

Changes made to the `/test` directory will not be seen by users. This allows free experimentation without affecting users' searches. If a serious error is made during customization of the `/test` directory files, simply copy the same file from the `/custom` directory and start over.

Once customizations are complete and satisfactory, copy the contents of `/test` into `/custom`. Click **RELOAD**. The new customized pages will now be accessible by users. Further customizations can be made in the same fashion by modifying the `/test` directory and copying its contents to the `/custom` directory.

The new `/test` and `/custom` are the directories that contain all your customizations for Inktomi Search Software's documents, but the original `docs` directory still has a significant function.



When you upgrade Inktomi Search Software, new documents are copied to the `/docs` directory, overwriting all its contents.

By keeping customized documents in the `/custom` directory, customizations will not be overwritten during an upgrade, nor will they be removed during an un-install. Also, the `/docs` directory acts as a final back-up for the `/custom` and `/test` directories.

## ■ Additional search presence

Sometimes, it is necessary to create more than one search presence for varying search needs. For example, perhaps the Widget Corporation would like to have a search interface that displays the Product, Pricing, and About the Widget Corporation collections, as well as pictures of widgets and a sales navigational bar. However, the Widget Corporation might also like to have a separate search interface for job recruiting, which would display the Job Listing, Benefits, and About the Widget Corporation collections, and also display graphics and navigational elements that are specific to their Human Resources department.

A consistent interface throughout initial search pages and search results pages can easily be achieved by creating additional copies of the `/docs` directory. New copies are nested within the `/docs` directory. Each of these nested directories can act as a separate search interface, and can be reached by adding the corresponding directory name to the end of the normal search URL.

Simply follow the instructions for creating a mirror of the `/docs` directory found in the previous segment, “Getting ready to customize” on page 47. Each presence will be under a separate directory nested within the active document directory (as displayed on the Parameters card in the Server view of the admin interface). Find out more about displaying different collections for different interfaces in [Limiting the collections displayed in the search box, on page 102](#).



## Customizing with Form Variables



You can customize the way in which Inktomi Search Software handles forms by manipulating its form variables.

This chapter contains the following information:

- [Changing the results page](#), on page **52**
- [Search specification](#), on page **53**
- [Search box display variables](#), on page **69**
- [Advanced Search variables](#), on page **71**
- [Implementing search form variables](#), on page **77**

## ■ Changing the results page

To change the look and content of the results page, create a stand-alone search form that passes the necessary form variables to the results page. A stand-alone search form is simply a search form that you can add to any page in your Web site. For more information, see [Creating a stand-alone search form](#), on page 101.

Inktomi Search Software uses the form variables defined in the `indexform_.html` or `queryform_.html` series (see [The `indexform\_.html` series](#), on page 27, and [The `queryform\_.html` series](#), on page 31) to build its default user interface. The values for each form variable in the `indexform_.html` or `queryform_.html` series call the values you set in the admin interface. For example, the “nh” parameter tells Inktomi Search Software to look at the **Server/Parameters** tab of the admin interface for the **Number of hits** parameter. You can modify the form variables in the `indexform_.html` or `queryform_.html` series, but usually it is better to set these variables in a stand-alone search form.

It is best to set as few form variables as possible. Many form variables (particularly those that represent default values set in the Query Parameters view of the admin interface) do not need to have values specified. See [Customizing with the Admin Interface](#) for more information.

For examples of how to set form variables, see [Implementing search form variables](#), on page 77. Inktomi Search Software’s form parameters are grouped according to search specification, search box display, and advanced search.

## ■ Search specification

Following are the parameters that specify search:

Specification	Description	Data type
<b>col</b>	Collections to query	string: internal name of collection
<b>qt</b>	Query text	search syntax string
<b>nh</b>	Number of hits to show	integer
<b>st</b>	Starting hit number	integer
<b>lk</b>	Look of the results pages	1 - show summaries 2 - hide summaries
<b>rf</b>	Results filter	0 - score by relevance 1 - score using date
<b>rq</b>	Requested query	0 - start a new search 1 - search these results 2 - search the Web
<b>oq</b>	Old query	string
<b>qp</b>	Query prefix	search syntax string
<b>qs</b>	Query suffix	search syntax string
<b>pw</b>	Page width	percentage or integer
<b>ws</b>	Word score	0 - do not show word score 1 - show word score
<b>ql</b>	Query level	a - advanced
<b>fs</b>	Find similar	URL from index
<b>ct</b>	Current topic	integer for topic ID
<b>ht</b>	Home topic	integer for topic ID
<b>si</b>	Search the Internet	0 - disabled 1 - enabled
<b>ex</b>	Extra query terms, as in thesaurus	string

### col

Set the `col` (collection) parameter to determine which collections are searched. Note that although the `col` parameter determines which collections are *searched*, the `qc` parameter determines which collections are *displayed*.

### Using “col” in Inktomi Search Software’s default interface

When you build your search interface using the `indexform_.html` and `queryform_.html` series, the `col` parameter looks for your setting in “Show this collection by default” on the **Collections/Tuning** tab.

By default, this parameter is populated with a bit of Python that automatically grabs internal collection names.

## Using “col” in a stand-alone search form

If you do not create a form element for `col` in your stand-alone search form, Inktomi Search Software will use your settings in the admin interface to determine whether a particular collection should be searched. If you do insert a form element for `col` in your stand-alone search form, the settings in the stand-alone search form will override those in the admin interface.

Sometimes it is necessary to set the `col` parameter in your stand-alone search form so that it differs from the setting in the admin interface. For example, suppose you had a collection that you did not want most users to search. However, you have a protected area of your intranet where certain users have access, and you want the restricted collection to be searchable in this area. In such a case, it makes sense to manually set the `col` variable in the stand-alone search form to enable the collection to be searchable, even though the admin interface is set to not search the collection by default.

To manually set the `col` parameter, therein limiting search to the collection designated, set the value of the `col` variable to the collection’s internal name:

```
<input type="hidden" name="col" value="widgets">
```

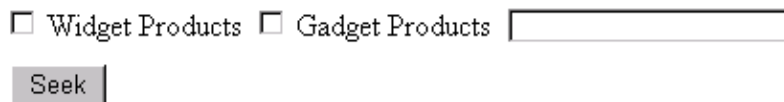
In the above setting, Inktomi Search Software will only provide search for the “widgets” collection. This is a hidden form element, meaning that the user has no control over the setting. To set multiple collections, separate each internal collection name with a space:

```
<input type="hidden" name="col" value="widgets gadgets">
```

In addition to the hidden form element, you can set the `col` variable in any of the usual form elements, such as checkbox, radio button, or listbox. These types of form elements give the user the option of searching the collection. For example, suppose the Widget Webmaster would like to create an initial search page where only the “widget” and “gadget” collections could be searched. Those two collections should be visible to the user, and the user can choose to search one or both of the collections. The Widget Webmaster decides to populate the `col` parameter as checkbox form elements within the `form` container tags:

```
<form action="http://[search server hostname_port]/query.html" method="GET">
<input type="checkbox" name="col" value="widget">Widget Products
<input type="checkbox" name="col" value="gadget">Gadget Products
<input type="text" size="30" maxlength="35" name="qt"><br>
<input type="submit" value=" Seek ">
</form>
```

FIGURE 26 Specifying which collections to query:



The image shows a web form with two checkboxes, one labeled "Widget Products" and one labeled "Gadget Products". To the right of these checkboxes is a text input field. Below the checkboxes and the input field is a button labeled "Seek".

However, if the `col` parameter is populated only on the initial search page, the user will still have the option of searching other collections in the search results pages.

If the intent is to only display and provide search for a limited number of collections, it is important to set both the `col` and `qc` parameters in the appropriate index and query documents. See [Limiting the collections displayed in the search box](#), on page 102 for more information.

You can also use the `col` parameter to determine the order in which your collections are listed. By default, Inktomi Search Software lists collection checkboxes in the search interface alphabetically. However, if you specifically set a list of collections using the `col` parameter, Inktomi Search Software will display them in the order in which you have listed your `col` form elements.

## qt

The `qt` (query text) parameter reflects the query the user types into the search box. By default, the search box is set to a length of 40 characters, but a user can type in up to 2033 characters.

Use the same format regardless of whether you are modifying files in the `indexform_.html` and `queryform_.html` series, or creating a stand-alone search form.

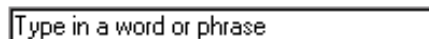
```
<input type=text name=qt size=40 value="&$qt;" maxlength=2033>
```

The Python value `&$qt;` tells Inktomi Search Software to take the information the user enters into the search box and send it to the search results pages. It is possible to change the size of the search box and the maximum length of allowed characters by adjusting the “size” and “maxlength” values.

By default, the search box is blank, waiting for a user to type a query. However, it is possible to add content in the search box by populating the `qt` parameter. For example, suppose the Widget Webmaster would like to have the search box display the suggestion, “Type in a word or phrase”. You can easily accomplish this by setting the `qt` value, either in your stand-alone search form or within the `indexform_.html` and `queryform_.html` series:

```
<input type=text name=qt size=40 value="Type in a word or phrase" maxlength=2033>
```

FIGURE 27 Adding a suggestion to the search box:



Instead of seeing a blank search box, users will now see a search box that prompts them to type in a word or phrase. However, `qt` is a good example of a parameter that does not need to be populated. Leaving the search box blank is usually the best course to take.

## nh

The `nh` (number of hits) parameter designates the number of hits to show per page in the search results. You do not need to set this parameter through the HTML source, instead you can easily adjust it in the **Server/Parameters** tab on the admin interface. Go to “Display number of hits” in the Query Parameters heading halfway down the card, and select the desired setting from the corresponding listbox.

However, you may choose to manually set this parameter in the HTML source if you have several customized search interfaces, where each interface displays a different number of hits in the corresponding results pages. Or, you may choose to set the number manually if you would like to provide the user with the ability to determine how many hits per page to display. Most of the time, however, it makes more sense to use the setting you designate in the admin interface.

## Using `nh` in Inktomi Search Software's default search interface

If you are using Inktomi Search Software's default search interface, where the initial search page is built through the `indexform_.html` series, look for the existing `nh` hidden form element in the `indexform_.html` page:

```
<input type=hidden name=nh value="&$nh;">
```

(For more information about the `indexform_.html` series, see [The `indexform\_.html` series](#), on page 27.) This is a hidden form element, meaning that the user has no control over the setting. The Python parameter "`&$nh;`" tells Inktomi Search Software to use the value for "Number of hits" that you have specified in the admin interface. To manually set the number of hits displayed on the results page, change the `value` attribute to an integer:

```
<input type=hidden name=nh value="25">
```

The above form element will tell Inktomi Search Software to return up to 25 hits on each results page, regardless of what is specified in the admin interface.

## Using "`nh`" in a stand-alone search form

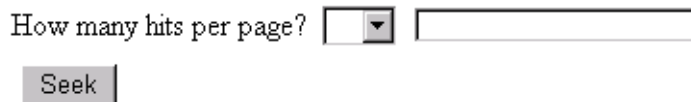
If your search interface is created through a stand-alone search form (see [Creating a stand-alone search form](#), on page 101), you may manually set this variable through any of the various form elements: hidden, listbox, or radio button. For instructions on how to integrate form variables into various form elements, see [Implementing search form variables](#), on page 77.

The most common way of customizing with the `nh` parameter is to put a listbox on the initial search page that allows the user to choose how many hits should appear in the search results. To do this, set the `value` attribute of the form element to equal the number of hits to display for each option:

```
<form action="http://[search server hostname and port]/query.html" method="GET">
<p>How many hits per page?
<select name="nh" >
<option value="10">10
<option value="25">25
<option value="50">50
</select></p>
<input type="text" size="30" maxlebngh="35" name="qt"><br>
<input type="submit" value=" Seek ">
</form>
```

In [Figure 28](#), a listbox lets the user choose whether to display 10, 25, or 50 hits per search results page.

**FIGURE 28** Specifying the number of hits:



How many hits per page?

## st

The `st` (starting hits) parameter designates the starting hit number Inktomi Search Software will display in the search results. By default this value is set to “1” in the `indexform_.html` and `queryform_.html` series, meaning that the first search result Inktomi Search Software displays is the first hit that Inktomi Search Software found for that query:

```
<input type="hidden" name="st" value="1">
```

Changing the starting hits number to 5, for example, will mean that Inktomi Search Software skips the first 4 hits for a query, and displays the 5th hit first. The user can still navigate back to the first 4 hits by clicking the 1-4 arrow in the top or bottom navigation bar.

To change the starting hits number, change the value attribute to an integer that reflects the desired starting hit number. However, there is rarely a situation that would require skipping first hits. Therefore, we do not recommend adjusting the `st` parameter.

## lk

The `lk` (look of the results page) parameter designates whether the results page should hide or show summaries by default. You do not need to set this parameter in the HTML. Instead you can adjust it on the **Server/Parameters** tab on the admin interface. Go to “Default look” in the Query Parameters heading halfway down the card, and select the desired setting from the corresponding listbox.

If for some reason you need to adjust the `lk` setting through the HTML, you can set it through form elements such as hidden, listbox, or radio button, depending on what type of options you want the user to have. For instructions on how to integrate form variables into various form elements, see [Implementing search form variables](#), on page 77.

### Using `lk` in Inktomi Search Software’s default search interface

Populating `lk` through a hidden form variable will designate the look of the results page without the user knowing it has been set. If using default search pages from the `indexform_.html` and `queryform_.html` series, look for the existing `lk` hidden form element:

```
<input type="hidden" name="lk" value="&$lk;">
```

See [The `indexform\_.html` series](#), on page 27 for more information about the `indexform_.html` series.

Change the value attribute to either 1 or 2, depending on whether you wish to hide or show summaries, respectively:

```
<input type="hidden" name="lk" value="1">
```

### Using “lk” in a stand-alone search form

If you are using a stand-alone search page (see [Creating a stand-alone search form](#), on page 101), add the above element to your form, adjusting the value attribute as needed:

```
<form action="http://[search server hostname and port]/query.html" method="GET">
<input type="hidden" name="lk" value="1">
<input type="text" size="30" maxlength="35" name="qt"><br>
<input type="submit" value=" Seek ">
</form>
```

## rf

The `rf` (results filter) parameter designates whether the results page score hits by date or by relevance. You do not need to set this parameter in the HTML, instead you can easily adjust it in Server Parameters of the admin interface.

Simply look under the Query Parameters heading halfway down the card, go to “Default results filter” and select the desired setting from the corresponding listbox.

If for some reason it is necessary to adjust the `rf` setting through the HTML, it can be populated through form elements such as hidden, listbox, or radio button, depending on what type of options the user should have. For instructions on how to integrate form variables into various form elements, see [Implementing search form variables](#), on page 77.

Using `rf` in Inktomi Search Software’s default search interface

Populating `rf` through a hidden form variable will designate the results filter without the user knowing it has been set. If using default search pages from the `indexform_.html` and `queryform_.html` series (see [The indexform\\_.html series](#), on page 27), look for the existing `rf` hidden form element:

```
<input type="hidden" name="rf" value="&$rf;">
```

Change the value attribute to either 0 or 1, depending on whether you wish to score by relevance or score by date, respectively:

```
<input type="hidden" name="rf" value="1">
```

### `rf` in a stand-alone search form

If using a stand-alone search page (see [Creating a stand-alone search form](#), on page 101), simply add the above element to your form, adjusting the value attribute as needed:

```
<form action="http://[search server hostname and port]/query.html" method="GET">
<input type="hidden" name="rf" value="1">
<input type="text" size="30" maxlength="35" name="qt"><br>
<input type="submit" value=" Seek ">
</form>
```

## rq

The `rq` (requested query) parameter designates whether Inktomi Search Software should search the query against one of the following:

- other collections (“Start new search”)
- a subset of the previous search (“Search these results”)
- the entire Web (“Search the entire Web”)

This parameter behaves similarly to the `qm` parameter (see “`qm`,” page 69).

### Using `rf` in Inktomi Search Software’s default search interface

On the default search pages built through the `indexform_.html` and `queryform_.html` series, `rq` by default is built into radio button form elements that allow users to either search among collections or search the Internet. (See [The indexform\\_.html series](#), on page 27 for more information about the

indexform\_.html series.)

```
<input type="radio" name="rq" value="0" checked onclick="setCols(true)">
<input type="radio" name="rq" value="2" onclick="setCols(false)">
```

FIGURE 29 Specifying the search collections or the Internet:

- ☒ My Collection
- ☐ The Internet

Each `rq` radio button form element contains a Python call that either extracts a list of available collections or ignores the collections altogether, depending upon which is selected. There is also a snippet of Python code that enables this behavior.

The `rq` parameter is the only parameter that is set by default in the form of radio buttons. This forces the user to either send a query against collections or search the Internet, but not both. If more than one collection is available, `rq` radio buttons also give the user more options. If they choose to search among collections, they are able to choose which collections to search in the form of checkboxes.

### Using `rq` in a stand-alone search form

Although Inktomi Search Software uses the `rq` parameter in radio buttons by default, you can set the parameter in any of the usual form elements: checkbox, listbox, radio, and hidden.

On initial search pages, there is no need for the “Search these results” option (value 1). However, it is possible to force a query subset by setting the collections `rq` radio button form element to “1”, and then setting a value for the `oq` parameter (see “`oq`,” page 59).

For example, suppose the Widget Webmaster would like to set the `rq` and `oq` parameters so that each query is a subset of a query for “widgets”:

```
<form action="http://[search server hostname and port]/query.html" method="GET">
<input type="hidden" name="rq" value="1">
<input type="hidden" name="oq" value="products">
<input type="text" size="30" maxlength="35" name="qt"><br>
<input type="submit" value=" Seek ">
</form>
```

Search results will be displayed as if the user had issued a query for “products”, selected “Search these results”, then entered another query:

```
products | [user's query]
```



NOTE

However, we don't recommend you do this. Instead, set the “`qp`” variable, described on [qp](#), on page 60.

### `oq`

The `oq` (old query) parameter is used with the “Search these results” option. When a user performs a

query, and then performs another query with “Search these results”, Inktomi Search Software designates the first query through the `oq` parameter.

By default, `oq` has an empty value:

```
<input type=hidden name=oq value="">
```

The `oq` value can be set to any string, using normal search syntax. On initial search pages, both the `oq` parameter and the `rq` parameter should be set. However, we do not recommend that you populate the `oq` variable.

## qp

The `qp` (query prefix) parameter is a form variable, usually hidden, that you can use to pre-filter the documents.

### Using `qp` in Inktomi Search Software’s default search interface

By default, `qp` has an empty value:

```
<input type=hidden name=qp value="">
```

The `qp` parameter allows you to create “virtual collections”, which are subsets of existing collections. The `qp` parameter is set to a value that is simple query syntax, which will then prefix the user’s query.

The content of the `qp` variable is prefixed to the user’s query with a double-pipe. The format is as follows:

```
[qp content] || [user’s query]
```

### Using `qp` in a stand-alone search form

To fully understand the `qp` form variable, you must understand how the pipe and double-pipe work in search syntax. See [Double-pipe](#), on page 125. Prefixing the query with the double-pipe creates a “Search these results” behavior. Submitting a query that contains a `qp` form variable is similar in behavior to submitting a query (`qp`), clicking “Search these results” (pipe), then submitting another query (user’s query).

For example, suppose the Widget Webmaster has a hidden form element that contains a `qp` variable with a value of `widget`:

```
<input type="hidden" name="qp" value="widget">
```

Now suppose a user types the word ‘price’ in the query box. Inktomi Search Software interprets the user’s query with the following syntax:

```
widget || price
```

The resulting documents must include the word `widget` and must include the word `price`.

The `qp` variable can be set to just about any string, in the same way that a regular query is made up of natural language. However, the better the search syntax in the `qp` variable, the better the results. See [Why proper search syntax is important](#), on page 116 for more information on building good search syntax.

For example, instead of setting `qp` to ‘widgets’, suppose you set the `qp` variable to ‘+widgets -gadgets +site:widgets.com’:

```
<input type="hidden" name="qp" value="+widgets -gadgets +site:widgets.com">
```

Again, the user types the word `price` in the search box. The query now has more specific meaning:

```
+widgets -gadgets +site:widgets.com || price
```

The resulting documents must include the word `widgets`, must not include the word `gadgets`, must originate from the `widgets.com` site, and must include the word `price`.

By setting a site name in the `qp` variable, a “virtual collection” is created for the site. This is a powerful tool when creating more than one search presence. Simply set a value for the `qp` variable on a customized search form, and the results are similar to having built a separate, unique collection.

For example, suppose you have three sites in a single collection:

```
http://www.abc.com/  
http://www.xyz.com/  
http://www.pdq.com/
```

Now suppose you only wanted to search against `xyz.com`. You would populate the form’s `qp` variable with “`site:xyz.com`”, resulting in a virtual collection that contains content only from `xyz.com` machines.

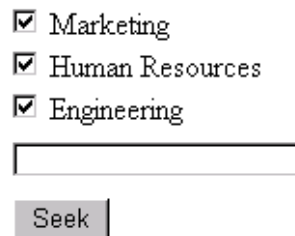
```
<input type="hidden" name="qp" value="site:xyz.com">
```

For an internal application, this might actually be machine names. For example, setting the `qp` value to “`site:otherhost.xyz.com`” would isolate search to content on that machine only.

In addition to hidden form elements, you can use `qp` in listboxes, single checkboxes, and radio buttons in your stand-alone search form:

```
<form action="http://[search server hostname and port]/query.html" method="GET">  
<input type="checkbox" name="qp" value="url:marketing" checked>Marketing<br>  
<input type="checkbox" name="qp" value="site:hr" checked>Human Resources<br>  
<input type="checkbox" name="qp" value="site:dev.widget.com" checked>Engineering<br>  
<input type="text" size="30" maxlength="35" name="qt"><br>  
<input type="submit" value=" Seek ">  
</form>
```

FIGURE 30 Specifying virtual collections:



☒ Marketing  
☒ Human Resources  
☒ Engineering

For more information on constructing forms using the `qp` variable, see [Implementing search form variables](#), on page 77.

## qs

The `qs` (query suffix) parameter appends search terms to the user’s query. By default, `qs` has an empty value:

```
<input type=hidden name=qs value="">
```

The `qs` parameter behaves in a similar way to the `qp` parameter, but with some very important differences. In each case, the user's query is appended with the content of the form variable, and the form variable takes a value that is in search syntax format (see [Why proper search syntax is important](#), on page 116).

In `qp`, search syntax is interpreted as `qp` content followed by a double-pipe, followed by the user's query:

```
[qp content] || [user's query]
```

In `qs`, content is simply appended to the end of the user's query, with no pipe or any other search operators.

```
[user's query] [qs content]
```

As a result of populating `qs`, the user's query is simply embellished with other search terms. There is no "Search these results" behavior unless the administrator designates it in the form of search syntax within the `qs`.

For example, suppose `qs` is populated with the words `widget`, `gadget`, and `thingamajig`:

```
<form action="http://search.widget.com/query.html" method="GET">
<input type="text" size="30" maxlength="35" name="qt">&nbsp;<br>
<input type="hidden" name="qs" value="widget gadget thingamajig">
<input type="submit" value="Seek">
</form>
```

Now suppose a user submits a query for the word `price`. The search engine receives a query that looks like this:

```
price widget gadget thingamajig
```

The corresponding search results will contain any or all of the above terms.

Like `qp`, `qs` becomes more valuable as better search syntax is introduced. For instance, suppose in the above example, the Widget Webmaster wanted to create a "Search these results" effect based on the user's query. The same above syntax can be used, with the addition of a pipe within the `qs` content:

```
<form action="http://search.widget.com/query.html" method="GET">
<input type="text" size="30" maxlength="35" name="qt">&nbsp;<br>
<input type="hidden" name="qs" value="| widget gadget thingamajig">
<input type="submit" value="Seek">
</form>
```

This time, if a user were to submit a query for the word `price`, the search engine would receive a query that looks like this:

```
price | widget gadget thingamajig
```

The corresponding search results will contain the word `price`, plus any or all of the words `widget`, `gadget`, or `thingamajig`.

The `qs` parameter is useful when set in conjunction with the minus operator, such as `"-site:widget.com"`. If certain content is to be excluded, it is better to set the minus operator in a `qs` rather than a `qp`.

## `qp` combined with `qs`

Used alone, `qs` is of little value; it is better to use the `qp` variable. However, using `qs` in conjunction with

`qp` and the proper search syntax can produce an extremely refined virtual collection.

For example, suppose the Widget Webmaster would like a page that restricts search to the two sites `widget.com` and `widget.co.uk`. Queries should also include either the word `widget`, `gadget`, or `thingamajig`. The `qp` and `qs` form variables would be set as follows:

```
<form action="http://search.widget.com/query.html" method="GET">
<input type="text" size="30" maxlength="35" name="qt">&nbsp;<br>
<input type="hidden" name="qp" value="site:widget.co.uk site:widget.com">
<input type="hidden" name="qs" value=" | Marketing HR Engineering ">
<input type="submit" value="Seek">
</form>
```

Corresponding search results will originate either from the `widget.com` site or the `widget.co.uk` site, will include whatever criteria the user sets in his query, and will also include either the word `widget`, `gadget`, or `thingamajig`. In the above example, queries are sent to the search engine in this format:

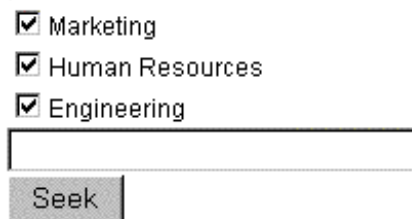
```
site:widget.com site:widget.co.uk || [user's query] | widget gadget thingamajig
```

Similar behavior can also be achieved by placing `qp` and `qs` in other form elements, such as checkboxes, listboxes, and radio buttons, depending on what types of choices the user should have.

For example, suppose the Widget Webmaster wanted to again restrict search to the `widget.com` and `widget.co.uk` sites, but this time give the user the option of choosing one of the products to be searched. One way to do this is to insert the words `Marketing`, `Human Resources` and `Engineering` in a listbox. This way, instead of search results containing a mixture of the three words, the user is given the option of choosing which product to search. This can be easily accomplished by putting `qp` in a hidden form element, and putting `qs` in a listbox:

```
<form name="seek" method="GET" action="http://search.widget.com/query.html">
<input type="text" size="30" maxlength="35" name="qt">&nbsp;<br>
<input type="hidden" name="qp" value="site:widget.co.uk site:widget.com">
<p>Which product would you like to search?</p>
<select name="qs" >
<option value="&$qs;">Marketing
<option value=" | widget">Human Resources
<option value=" | gadget">Engineering
</select>
<input type="submit" value="Seek">
</form>
```

FIGURE 31 Combining `qp` with `qs`:



☒ Marketing  
☒ Human Resources  
☒ Engineering

Notice the first option allows the user to search any product. The value is set to `&$qs;` which is the default

value for `qs`, leaving the variable empty.

In the above example, suppose the user submits a query for the word price, and selects Gadgets from the listbox. Search results will be returned using the following syntax:

```
site:widget.com site:widget.co.uk || sale price | gadget
```

The corresponding search results will originate from either `widget.com` or `widget.co.uk`, will include either the word `sale` or the word `price`, and will include the word `gadget`.

## pw

The `pw` (page width) parameter designates the expanse of the search pages within the browser window. You do not need to set this parameter in the HTML; instead you should set it through the **Server/Parameters** tab on the admin interface. Go to “Results page width” under the Query Parameters heading halfway down the card, and enter the setting from the corresponding text box.

Whether you set the page width through the admin interface or set it as a `pw` form variable, the value should be in the form of an integer or a percent. An integer setting will designate page width by exact number of pixels. A percentage setting will designate what percent of the browser window should be used by the search results page.

By default, the “`pw`” parameter is found in the opening “`table`” tag:

```
<table cellpadding=3 cellspacing=0 width="&$pw;">
```

If you wish to call the automatic page width into your custom HTML page, use the above format.

## ws

The `ws` (word score) parameter designates whether or not Inktomi Search Software should display word scores in the search results. Word scores appear next to each hit, and report how many times a term was counted within that document.

Do not set the `ws` parameter in the HTML; instead, set the parameter in the Server/Parameters tab in the admin interface. Go to the “Show individual word scores” checkbox under the Query Parameters heading halfway down the card.

### Using `ws` in Inktomi Search Software’s default search interface

By default, the `ws` parameter is in the form of a hidden form element:

```
<input type=hidden name=ws value="&$ws;">
```

### Using `ws` in a stand-alone search form

If using a stand-alone search form (see [Creating a stand-alone search form](#), on page 101), use the above format to use the same word score designation you have specified in the admin interface:

```
<form action="http://search.widget.com/query.html" method="GET">
<input type="text" size="30" maxlength="35" name="qt">&nbsp;<br>
<input type=hidden name=ws value="1">
<input type="submit" value="Seek">
</form>
```

If you need to populate the `ws` form element through the HTML, set the value to either “0” or “1”, depending on whether the word scores should be hidden or displayed, respectively.

## q1

The `q1` (query level) parameter determines whether the default search form should be simple or advanced.

Do not set the `q1` parameter in the HTML; instead, adjust its value in the Server/Parameters tab on the admin interface. Go to the “Default query form complexity” listbox under the Query Parameters heading halfway down the card.

### Using `q1` in a stand-alone search form

The `q1` parameter can come in the form of a URL:

```
<form action="http://search.widget.com/query.html" method="GET">
<input type="text" size="30" maxlength="35" name="qt">&nbsp;<br>
<input type="submit" value="Seek">
<a href="[search URL:port]?col=&ht=0&qc=&si=1&q1=a">Advanced</a>
<a href="[search URL:port/help/]>Help</a></form>
```

FIGURE 32 Specifying the query level:



## fs

The `fs` (find similar) parameter sends a URL to the search engine, and the search engine in turn finds documents that have content similar to that URL.

When the user clicks the “find similar” link, Inktomi Search Software uses the URL instead of a text query.

The `fs` parameter is another example of a form variable that you should not set manually in the HTML. However, setting an `fs` value is another way that a certain variation of a virtual collection can be made. Using `fs` in conjunction with a user’s query (`qt`) can produce a very refined set of results. In such a case the `qt` content acts strictly as refinement, not the base of a query.

### Using `fs` in a stand-alone search form

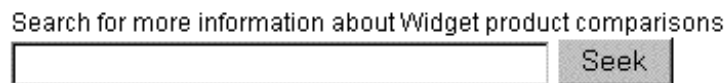
The `fs` parameter can be used in form elements such as hidden, single checkboxes, listboxes or radio buttons, or it can be used in the form of a hyperlink. When used within a form element, `fs` should be set to the URL of a document within the index. The URL MUST be in the index for `fs` to work. To verify whether a URL is in the index, run a query for the URL in URL Status of the help interface.

One way to implement `fs` into a form is to create a hidden form element. For example, suppose the

Widget Webmaster has a page in the index that compares Widgets to competitors' products. The Webmaster would like to have a specialty search box that finds pages similar to the comparison page, and also finds pages based on the user's query.

```
<form name="seek" method="GET" action="http://search.widget.com/query.html">
  <input type="hidden" name="fs" value="http://www.widget.com/compare.htm">
<p>Search for more information about Widget product comparisons:
  <input type="text" name="qt" size=40 value="" maxlength=2033>
  <input type="submit" value=" seek "></p>
</form>
```

FIGURE 33 Finding similar results:



In the above example, search results will be based on both the user's query and documents that contain content similar to `compare.html`.

However, be careful when creating specialized search pages such as this. It may not be clear enough to the user that his search will be based on Widget comparisons. A user may find what he needs on the comparison page, then type in a new search in the search box, without reading that his search will be based on comparisons only. This can result in some confusion.

A better solution is to use `fs` in the form of a hyperlink. You construct an `fs` hyperlink in query string format. For example, suppose in the above situation, the Widget Webmaster decides not to put a search box on the comparison page. Instead, s/he adds an `fs` hyperlink to the page's navigational link margin:

```
<a href="default.htm">Widget Home</a>
<a href="http://www.widget.co.uk/compare.htm">UK Product Comparisons</a>
<a href="http://search.widget.com/query.html?fs=http%3A//www.widget.com/
compare.htm">Browse Comparison Info</a>
```

When a user clicks the **Browse Comparison Info** hyperlink, a search results page appears that lists hits that have content similar to the `compare.htm` page.

When you set `fs` as a hyperlink, the first hit in the search results page will always be the page you set in the `fs` value. In the above example, this is the `compare.htm` page. You can have the search results skip the page set in the `fs` value by sending a starting hits `st` value along with the `fs` value.

For example, suppose the Widget Webmaster has composed an `fs` hyperlink to the `compare.htm` `fs` search results set. The Widget Webmaster wants Inktomi Search Software to skip the `compare.htm` hit, so that the user doesn't go back and forth in a loop between the `compare.htm` page and the search results. This can be accomplished by setting an `st` value that starts at hit number two, skipping the predictable first hit:

```
<a href="http://search.widget.com/query.html?fs=http%3A//www.widget.com/
compare.htm&st=2">Browse Comparison Info</a>
```

Inktomi Search Software passes the `st` parameter, appending `&st=2` to the end of the hyperlink. For

more information about the `st` parameter, see "st," page 57.

## ct

The `ct` (current topic) parameter specifies the current topic ID number. Inktomi Search Software CCE uses `ct` for topic click-throughs. If you do not have CCE, you will not have a use for `ct`.

You set the value for `ct` to the topic ID number of the topic. You can find out what the topic ID number is for a specific topic by clicking the link for that topic in the “Current topics” section of Topics Status in the admin interface.

Like `fs`, you can use `ct` in the form of a hyperlink. A `ct` hyperlink is constructed in query string format. For example, suppose you want to create a static link in the navigational margin that always points to the Widget topic. You know that the topic ID for the Widget topic is 370942204. You could add this hyperlink to the rest of the links in the navigational margin of your custom page:

```
<a href="default.htm">Widget Home</a>
<a href="http://www.widget.co.uk/compare.htm">UK Product Comparisons</a>
<a href="http://search.widget.com/query.html?ct=370942204">Browse Widget Info
</a>
```

When a user clicks the Browse Widget Info hyperlink, a search results page appears that lists the contents of the Widget topic.

## ht

You can use the `ht` (home topic) variable to specify an alternate browsing root of the CCE topic tree. This is useful if you would like to provide a topic browsing interface to a sub-tree of your entire topic tree. The `index.html` and `query.html` files support the `ht` form variable.

Using `ht` in a stand-alone search form

The default value of `ht` is 0, which is the topic id of the root of the CCE topic hierarchy. By explicitly specifying `ht` with an alternate topic id, you can provide an interface that limits “browse topics” and “related topics” to subtopics of your alternate. The `ht` form variable is most useful when used in combination with the `qp`, `col`, and `qc` form variables to provide a complete search and directory interface to a subset of your content.

By default, the `ht` parameter is in the form of a hidden form element:

```
<input type=hidden name=ht value="0">
```

### Using `ht` in a stand-alone search form

You can set the `ht` parameter in the form of hidden, checkbox, listbox, and radio button form elements in your stand-alone search form. In the example below, `ht` is in the form of a hidden form element.

```
<form action="http://search.widget.com/query.html" method="GET">
<input type="text" size="30" maxlength="35" name="qt">&nbsp;&nbsp;&nbsp;<br>
<input type=hidden name=ht value="0">
<input type="submit" value="Seek">
</form>
```

## si

The `si` (search the Internet) parameter determines whether Inktomi Search Software should provide a “Search the Internet” option. If this variable is non-zero, then an option to search the Internet is presented in the search forms. The default value of `si` is set in the “Provide option to search the Internet” setting in the “Query Parameters” section on the **Server/Parameters** tab on the admin interface.

### Using `si` in a stand-alone search form

The `index.html` and `query.html` files support the `si` form variable.

An explicit value of this variable is useful to override the default “Provide option to search the Internet” setting. For example, enable the option so your global intranet search page provides the option; you can also provide a more narrow site search facility with the option disabled, simply by having the URL of the site search form explicitly specify an `si` value of 0.

```
<form action="http://search.widget.com/query.html" method="GET">
<input type="text" size="30" maxlength="35" name="qt">&nbsp;<br>
<input type=hidden name=si value="0">
<input type="submit" value="Seek">
</form>
```

## ex

The `ex` (extra) form variable is the parameter Inktomi Search Software uses to pass thesaurus terms. Inktomi Search Software appends the terms in the “value” attribute of this form variable to the user’s query. Multiple terms are appended with commas to the query text.

### Using `ex` in a stand-alone search form

There are few situations where it makes sense to manually set the `ex` variable. Expanding the thesaurus is a better solution. To learn more about Inktomi Search Software’s built-in thesaurus, see Chapter 5 of the Inktomi Search Software Administrator’s Guide, *Administering the Server*.

The `query.html` file supports the `ex` form variable. You can manually add thesaurus checkboxes to your search form through the `ex` variable. For example, suppose you wanted to force a thesaurus on the word “announcements”. You would set the `ex` variable as follows:

```
<form action="http://search.widget.com/query.html" method="GET">
<input type="text" size="30" maxlength="35" name="qt">&nbsp;<br>
<input type=checkbox name=ex value="announcements">announcements<br>
<input type="submit" value="Seek">
</form>
```

The user then has the option of including the word “announcements” with his search, simply by checking the checkbox. But again, it is better to expand the built-in thesaurus than to manually set `ex` parameters.

## ■ Search box display variables

Following are the parameters that control the query box display:

Specification	Description	Data type
<b>qm</b>	Query mode	0 - start new search 1 - search these results 2 - search the Web
<b>qc</b>	Collections to list in the query box	string: internal name of the collection

### qm

The **qm** (query mode) variable determines whether the display should show “Start new search”, “Search these results”, or “Search entire Web”. This parameter is very similar in behavior to the **rq** parameter. The basic difference is that **rq** specifies search, whereas **qm** specifies display. When you use these two variables together, set them to the same value.

### qc

The **qc** (query collection) variable allows you to designate which collections Inktomi Search Software should display by the search box. This parameter is very similar to the **col** parameter. The basic difference is, **col** specifies search, whereas **qc** specifies display. The two parameters are usually used together to create search forms that are limited to a subset of collections. Both the **col** and the **qc** values should be set to internal collection names.

### Using qc in Inktomi Search Software’s default search interface

If the initial search page is built from the `indexform_`.html series, follow instructions to limit the number of collections searched. See [Limiting the collections displayed in the search box](#), on page 102 for more information.

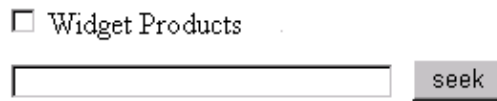
### Using qc in a stand-alone search form

If you are creating a static, custom search form (see [Creating a stand-alone search form](#), on page 101), you can control the display of collections throughout the initial search and search results.

For example, suppose the Widget Corporation Webmaster has built three collections: Product, International, and Intranet. On an external search page, the Webmaster wants to display and enable search for the Product and International collections, but not the secure Intranet collection. The Webmaster can easily accomplish this by inserting **qc** form elements within the `form` container tags:

```
<form name="seek" method="GET" action="http://search.widget.com/query.html">  
<input type="hidden" name="qc" value="product intl"> Widget Products  
<input type="text" name="qt" size=40 value="" maxlength=2033>  
<input type="submit" value=" seek "></p>  
</form>
```

**FIGURE 34** *Specifying each box display variables:*



☐ Widget Products

seek

See [Implementing search form variables](#), on page **77** for examples of how form variables can be inserted into various form elements.

You can also use the `qc` variable when maintaining security. Displaying and searching only certain collections is a way to restrict content. For more information on restricting the collections displayed in the search box, see [Limiting the collections displayed in the search box](#), on page **102**.

## ■ Advanced Search variables

To fully understand how Advanced Search form variables function, see [Why proper search syntax is important](#), on page 116.

In many cases, you can pass Advanced Search form variables only to Advanced Search forms. That is, if the initial search page is a custom page that uses Advanced Search parameters, the search results page must also be an Advanced Search page, although there are exceptions.

Following are the parameters that control Advanced Search:

Specification	Description	Data type
<b>op</b>	operator	+ required - excluded
<b>fl</b>	Field search	string
<b>ty</b>	type	w - words p - phrase n - name
<b>tx</b>	query text	string
<b>dt</b>	date range	an - anytime in - in the... ba - before or after
<b>la</b>	language	specified string

Each variable has a suffix of 0, 1, or 2, indicating whether the information is coming from the first, second, or third groups of the Advanced Search form. These parameters are used in `coreforma.html` in the `/docs` directory.

When modifying parameters in `coreforma.html`, it is important to establish all parameters within a set. For example, if you modify “`op0`”, it is important to also have a value for “`fl0`”, “`ty0`”, and “`tx0`”, even if that value is empty.

### op

The `op` (operator) parameter adds syntax in the form of search operators to the query.

In the Advanced Search form, the `op` variable is built into a listbox that has three options: “should contain”, “must contain”, and “must not contain”. For the “should contain” option, the `op` value is empty. For the “must contain” option, the `op` value is set to the plus sign, which requires a term. For the “must not contain” option, the `op` value is set to the minus sign, which excludes from the search results content containing that term.

There are three iterations of the `op` variable written in `coreforma.html` to match the three iterations on the Advanced Search form. [Figure 35](#) shows the snippet that matches group 1.

FIGURE 35 op1 code snippet

```
<select name=op1>
<!--$
op1 = query.get("op1",["+"])[0]
for val,nam in [("", "should contain"),
                ("+", "must contain"),
                ("- ", "must not contain")]:
    if op1==val: --><option value="&$val;" selected><!--$
    else: --><option value="&$val;"><!--$
    -->&$nam;
</select>
```

Of the three iterations of the op parameter, group 0 does not have a “must not contain” option. This is because a user should not use a minus operator unless there are more than one query groups used.

## f1

The f1 (field) parameter allows the query to mimic a field search. For more information on how field searches work, see [Field searches](#), on page 119.

By default, the f1 variable can be set to title, URL, site, link, alt, description, or keywords. More values for f1 can easily be added in the “Advanced query fields” text box at the bottom of the **Server/Parameters** tab in the admin interface.

For example, suppose there were many documents on the Widget Corporation’s Web site that contained variations of the following Meta tag:

```
<meta name="manufacturer" content="Widgetmania">
```

Under “Advanced query fields”, simply add the following text:

```
manufacturer:, in the manufacturer
```

As a result, the option “in the manufacturer” will be added to the corresponding drop-down list on the Advanced Search form.

Like most other Advanced Search parameters, f1 is expressed in coreforma.html in the /docs directory. Following are the default f1 settings from group 0:

```
<!--$
f10 = query.get("f10",[""])[0]
for val,nam in config.advanced_query_fields:
    if f10==val: --><option value="&$val;" selected><!--$
    else: --><option value="&$val;"><!--$
    -->&$nam;
</select>
```

## ty

The ty variable indicates what type of text is in the query box. This can be set to w, p, or n, depending on whether the query text should be interpreted as words, a phrase, or a proper name, respectively.

w, or “words”, is the default setting and does not affect the query. P, or “phrase” causes the corresponding text to be enclosed in quotation marks, which treats it as a phrase search. N, or “name” specifies the corresponding text as a proper name. Inktomi Search Software automatically capitalizes the first letter of each word to treat it as a proper name search.

Like most other Advanced Search parameters, ty is expressed in `coreforma.html` in the `/docs` directory. Following are the default ty settings from group 0:

```
<select name=ty0>
<!--$
ty0 = query.get("ty0",["w"])[0]
for val,nam in [("w","the words"),
                ("p","the phrase"),
                ("n","the name")]:
    if ty0==val: --><option value="&$val;" selected><!--$
    else: --><option value="&$val;"><!--$
    -->&$nam;
</select>
```

## tx

The tx (text) parameter refers to the query text associated with each group. This is the text that the user types into the text box. The tx parameter functions in a way similar to the qt parameter in the Simple Search form.

Like most Advanced Search form variables, qt must be tied to a group. This way, Inktomi Search Software can interpret the use of an operator, field search, or naming property to be tied to a string of query text. Following are the default tx settings from group 0:

```
<!--$
tx0 = string.join(query.get("tx0",[""]))
--><input type=text name=tx0 size=50 value="&$tx0;" maxlength=512>
```

## dt

The dt parameter represents the date range module. This can be set to an, in, or ba, depending on whether the query should be searched within any time period, in a time period restricted to the last day or the last week, or within a “before or after” time period, respectively.

The value setting of an is the default. This value means “anytime”, so the corresponding query will be searched in all time periods.

The value setting of in means “in the...[restricted time period]”, and allows users to restrict the query to the last day or the last week. This value will be grabbed from the parameter “inthe”, which by default is expressed in a listbox. The “inthe” parameter is then set to a value in seconds, which tells Inktomi Search Software how many seconds into the past it should look for document dates when creating search results. For example, a value setting of 86400 will tell Inktomi Search Software to look only for documents which are dated within the last 24 hours. A value setting of 604800 will tell Inktomi Search Software to look only for documents which are dated within the last week.

Like most other Advanced Search parameters, dt is expressed in `daterange.html` in the `/docs` directory. [Figure 36](#) shows the default dt settings for in and an:

FIGURE 36

```

<!--$
dt = query.get("dt",["an"])[0]
if dt=="an": --><input type=radio name=dt value=an checked><!--$
else: --><input type=radio name=dt value=an><!--$
--></td>
<td>&$self.text(u'Anytime');</td>
</tr>
<tr>
<td><p><!--$
if dt=="in": --><input type=radio name=dt value=in checked><!--$
else: --><input type=radio name=dt value=in><!--$
--></td>
<td colspan=2>
<select name=inthe onchange="document.&$formname;.dt[1].checked=true">
<!--$
try: inthe = self.atoi(query["inthe"][0])
except: inthe = 86400*7
for i,m in [(86400*7,u"in the last week"),
            (86400*14,u"in the last 2 weeks"),
            (86400*30,u"in the last 30 days"),
            (86400*60,u"in the last 60 days"),
            (86400*90,u"in the last 90 days"),
            (86400*180,u"in the last 180 days"),
            (86400*365,u"in the last year"),
            (86400*365*2,u"in the last 2 years")]:
    if i==inthe: --><option value=&$i; selected><!--$
    else: --><option value=&$i;><!--$
    -->&$self.text(m);
</select>

```

The value setting of “ba” tells Inktomi Search Software to look *before* or *after* a certain date. On the Advanced Search form, this is expressed in listbox format. The “ba” variable may be associated with any one of the following variables:

- amo - after month
- ady - after day
- ayr - after year
- bmo - before month
- bdy - before day
- byr - before year

Unlike most advanced search form variables, the “dt” parameter and its corresponding variables will work regardless of whether they are embedded in a simple or an advanced query form. This enables you to customize a date range search on a simple search form. [Figure](#) shows the default “dt” settings for “ba”:



The **[same line]** indicator in the snippets means that line of code should be on the same line as the line above it. We are unable to properly display the code due to formatting restrictions. Spacing is important.

```

<!--$
if dt=="ba": --><input type=radio name=dt value=ba checked><!--$
else: --><input type=radio name=dt value=ba><!--$

```

```

--></td>
<td align=right>&$self.text(u'on or after');</td>
<td>
<!--$
try: amo = self.atoi(query["amo"][0])
except: amo = time.localtime(self.now-86400*6)[1]
try: ady = self.atoi(query["ady"][0])
except: ady = time.localtime(self.now-86400*6)[2]
try: ayr = self.atoi(query["ayr"][0])
except: ayr = time.localtime(self.now-86400*6)[0]
-->
<input name=ady size=2 maxlength=2 value=&$ady;
onchange="document.&$formname;.dt[2].checked=true">
<select name=amo onchange="document.&$formname;.dt[2].checked=true">
<!--$
for i,m in [(1,u"January"),
            (2,u"February"),
            (3,u"March"),
            (4,u"April"),
            (5,u"May"),
            (6,u"June"),
            (7,u"July"),
            (8,u"August"),
            (9,u"September"),
            (10,u"October"),
            (11,u"November"),
            (12,u"December")]:
    if i==amo: --><option value=&$i; selected><!--$
    else: --><option value=&$i;><!--$
    -->&$self.text(m);

</select>
<input name=ayr size=4 maxlength=4 value=&$ayr;
[same line]onchange="document.&$formname;.dt[2].checked=true">
</td>
</tr>
<tr>
<td></td>
<td align=right>&$self.text(u'and before');</td>
<td>
<!--$
try: bmo = self.atoi(query["bmo"][0])
except: bmo = time.localtime(self.now+86400)[1]
try: bdy = self.atoi(query["bdy"][0])
except: bdy = time.localtime(self.now+86400)[2]
try: byr = self.atoi(query["byr"][0])
except: byr = time.localtime(self.now+86400)[0]
-->
<input name=bdy size=2 maxlength=2 value=&$bdy;
onchange="document.&$formname;.dt[2].checked=true">
<select name=bmo onchange="document.&$formname;.dt[2].checked=true">
<!--$
for i,m in [(1,u"January"),
            (2,u"February"),
            (3,u"March"),
            (4,u"April"),
            (5,u"May"),
            (6,u"June"),
            (7,u"July"),

```

```

        (8,u"August"),
        (9,u"September"),
        (10,u"October"),
        (11,u"November"),
        (12,u"December")]]:
    if i==bmo: --><option value=&$i; selected><!--$
    else: --><option value=&$i;><!--$
    -->&$self.text(m);
</select>
<input name=byr size=4 maxlength=4 value=&$byr;
    [same line]onchange="document.&$formname;.dt[2].checked=true">

```

## 1a

The 1a parameter represents the language module. If your license key enables multiple languages, a language listbox appears on the Advanced Search forms. The 1a parameter can be set to any of the following:

- English - en
- Dutch - du
- Spanish - sp
- French - fr
- German - de
- Italian - it
- Portuguese - po
- Swedish - sw
- Danish - da
- Finnish - fi
- Norwegian - no
- Japanese - ja
- Korean - ko
- Simplified Chinese - zh\_cn
- Traditional Chinese - zh\_tw

Unlike most advanced search form variables, the 1a parameter will work regardless of whether it is embedded in a simple or an advanced query form. This enables you to customize a language search on a simple search form.

We strongly recommend that any manual modifications to the 1a parameter be avoided, as it will cause problems with various items in the user interface. We do not provide support for modifications to the 1a parameter.

## ■ Implementing search form variables

The parameters described in "Changing the results page," page 52 can easily be integrated into search forms. The most commonly customized variables are `qp` and `qc`.

The subsequent sections discuss form variable integration into hidden, checkbox, listbox, and radio button formats. Each format has its own usage and follows a separate syntax. However, the following rules apply to all of Inktomi Search Software's parameters, regardless of which format is used:

- Each form variable will have a value that requires a setting of either a string, an integer, or a preset value, depending upon which form variable is used. When the "value" attribute requires a string, multiple terms are space-delimited.
- When working with form variables, it is important to be mindful of table structure. Inktomi Search Software's forms are usually integrated closely with tables. See [Tables](#), on page 26 for a description of table structure.
- When creating custom search pages, make sure your form elements occur within the "form" container tags that point to Inktomi Search Software. Otherwise, they will be ignored.
- Be careful of multiple occurrences of the same form variable. Many form variables can only be sent to the search results in a single iteration. In the HTML source for your search page, look for a pre-existing occurrence of the form variable you intend to populate. If the form variable already exists on the page, modify the existing form element, or remove it altogether and start anew. The only form elements that can have multiple iterations are `qc`, `col`, and `qs`.

### Hidden form elements

Hidden form elements allow you to add search criteria without affecting the user interface. The user has no option of enabling each element's use; they are simply passed to the search results with no user interference. Hidden form elements can be integrated with Inktomi Search Software's form variables using the following syntax:

```
<input type="hidden" name="[parameter]" value="[value]">
```

In the following example, a hidden form element is added to a custom search page using the `qp` variable:

```
<form name="seek" method="GET" action="http://search.widget.com/query.html">
<input type="hidden" name="qp" value="+site:widget.com widget gadget">
<input type="text" name="qt" size=40 value="" maxlength=2033>
<input type="submit" value=" seek ">
</form>
```

In the above example, search results will be restricted to the `widget.com` site, and may contain the word `widget` and/or the word `gadget`.

For the `qp` variable, the "value" attribute is populated with a string that should mimic search syntax. Multiple terms are space-delimited within the "value" attribute.

### Integrating variables into checkboxes

Checkboxes provide the user with "and/or" choices. When submitting a form with checkboxes, users may select any or all of the options provided.

Checkboxes usually come in multiples, and take the following syntax:

```

☐

```

In the above sequence, three checkboxes are created. Adding the “checked” attribute into the checkbox form element will cause that checkbox to be checked by default. However, the user can still toggle the checkbox on or off.

The `col` form variable is a good example of a parameter that can be integrated into multiple checkboxes.

For example, suppose the Widget Corporation has four collections with the following internal collection names: `corp`, `products`, `intl`, and `pub`. The Webmaster has made a custom initial search page, and would like to list these collections for search. The Webmaster chooses to associate each collection with a checkbox, so that users may select which departments to search:

```

<form name="seek" method="GET" action="http://search.widget.com/query.html">
<b>Search the collections:</b><br>
<input type="checkbox" name="col" value="corp" checked>Corporate
<input type="checkbox" name="col" value="products" checked>Products
<input type="checkbox" name="col" value="intl" checked>International
<input type="checkbox" name="col" value="pub" checked>Publications
<br>
<input type="text" name="qt" size=40 value="" maxlength=2033>
<input type="submit" value=" seek ">
</form>

```

FIGURE 37 *Checkbox form elements:*

**Search the collections:**

☒ Corporate ☒ Products ☒ International ☒ Publications

The users may now chose to search “Corporate” and “International”, or “Products” and “Corporate”, or whatever combination of these collections suit them. Upon submission of the form, the user will get results from collections matching the selected checkboxes.

In the preceding example, the form variables are processed by Inktomi Search Software and passed as a URL:

```
http://search.widget.com/query.html?col=corp&col=products&col=intl&col=pub
```

Form variables such as `col` work easily within checkboxes, because more than one iteration of `col` can be passed to the search results page. However, not all form variables can be passed in multiple iterations. Form variables that are easily integrated into checkboxes are `col`, `qp`, and `qc`. The `qs` form variable can also be integrated into checkboxes, but such an integration is not very useful.

## Integrating form variables into a listbox

A listbox is a useful form element when the user is to make an “either/or” selection. Listbox form elements can be integrated with Inktomi Search Software’s form variables using the following syntax:

```

<select name="[parameter]">
<option value="[value]" default>[Label1]
<option value="[value]">[Label2]
<option value="[value]">[Label3]
</select>

```

In the above sequence, three options will appear in the listbox. Notice that only one of those options has been pre-selected, indicated by the “default” attribute. This is because only one option can be selected with listboxes.

Suppose the Widget Corporation’s content is divided into departments. Widget’s Webmaster could add a listbox to the search form that would allow users to choose a department. In this example, the `qp` variable is used so that each option acts as a virtual collection:

```

<form name="seek" method="GET" action="http://search.widget.com/query.html">
<b>Search the departments:</b>
<select name="qp">
<option value="" selected>Any department
<option value="site:hr.widget.com">Human Resources
<option value="site:mktg.widget.com/">Marketing
<option value="url:sales">Sales
<option value="url:research">Research
</select><br>
<input type="text" name="qt" size=40 value="" maxlength=2033>
<input type="submit" value=" seek ">
</form>

```



The user may select only one of the options provided in the listbox.

FIGURE 38 Form variables presented in listboxes:

## Integrating form variables into radio buttons

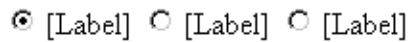
Radio buttons produce behavior similar to listboxes, in that the user is given an “either/or” choice. Radio button form elements can be integrated with Inktomi Search Software’s form variables using the following syntax:

```

<input type="radio" name="[parameter]" value="[value]" checked>[Label]
<input type="radio" name="[parameter]" value="[value]">[Label]
<input type="radio" name="[parameter]" value="[value]">[Label]

```

FIGURE 39 Specifying radio buttons:



In the above sequence, three radio buttons are created. Notice that only one of those buttons has been pre-selected, indicated by the “checked” attribute. This is because, with radio buttons, only one option can be selected at a time.

Suppose the Widget Corporation has three search categories to offer in one search interface: Products, Company Information, and Publications. These categories are created in the form of `qp` virtual collections.

The user could search one of these categories by selecting the appropriate radio button. On the custom search form, the radio form elements are listed as follows:

```
<form name="seek" method="GET" action="http://search.widget.com/query.html">
<h3>Search:</h3>
<input type="radio" name="qp" value="widget gadget" checked>Widget Products<br>
<input type="radio" name="qp" value="site:widget.com">Widget Company<br>
<input type="radio" name="qp" value="url:publications">Widget Publications<br><br>
<input type="text" name="qt" size=40 value="" maxlength=2033>
<input type="submit" value=" seek ">
</form>
```

FIGURE 40 Form variables presented in radio buttons:

A screenshot of a web search form. The form is titled "Search:" in bold. Below the title, there are three radio buttons with labels: "Widget Products", "Widget Company", and "Widget Publications". The "Widget Products" radio button is selected. Below the radio buttons, there is a text input field and a "search" button.

## Integration



Integrating Inktomi Search Software with your organization's look and feel may mean integrating frame sets or customizing Inktomi Search Software's style sheet.

This section contains the following information about Inktomi Search Software integration:

- [Style sheets](#), on page **82**
- [Calling Inktomi Search Software's style sheet](#), on page **82**
- [default.css](#), on page **82**
- [sample.css](#), on page **83**
- [Managing frame sets](#), on page **84**
- [Sending results pages to a surrogate frame set](#), on page **84**

## ■ Style sheets

Inktomi Search Software integrates style sheets with its document set. You can use style sheets to easily set style formats for your Web interface. Style sheets allow you to define fonts, colors, and other style elements.

How to create and modify style sheets is beyond the scope of this document. The World Wide Web Consortium (W3C) defines the standards for style sheets. You can read more about the W3C style sheet standards online at:

<http://www.w3.org/TR/WD-html40-970708/present/styles.html>

## Calling Inktomi Search Software's style sheet

By default, Inktomi Search Software calls the file `default.css` as its style sheet. This style sheet is called in the `link` tag of `header.html`:

```
<link rel=stylesheet href="default.css">
```

If you want Inktomi Search Software to use your custom style sheet, change `header.html` to reference your style sheet. For example, suppose you had a custom style sheet in your active document directory that you named `newstyle.css`. You would change the link relation in `header.html` as follows:

```
<link rel=stylesheet href="newstyle.css">
```

## default.css

The file `default.css` defines all basic style elements that Inktomi Search Software will need to create its user interface. If you choose to modify this file, be sure to make a back-up to preserve the original format. For more information on backing up, see [Getting ready to customize](#), on page 47.

The file `default.css` begins with comments, then lists style elements:

```
/* Basic style */

/* BODY {font-family: Georgia, "Times New Roman", Times, "New York", serif;} */

/* This is here to get the message through to Navigator 4.x */
/* P {font-family: Georgia, "Times New Roman", Times, "New York", serif;} */

/* This is here to get the message through to Navigator 4.x */
/* TD {font-family: Georgia, "Times New Roman", Times, "New York", serif;} */

.query {
    background-color: white;
    border-color: black;
}

.tip {
    font-style: italic;
    font-size: small;
    background-color: #C0C0C0;
    color: black;
}
```

```
DIV.results .highlight {  
    background-color: #FFFF99;  
}  
  
DIV.results .wordscores {  
    font-size: x-small;  
    color: #666666;  
}
```

## sample.css

Use the file `sample.css` as an aid when you do your style sheet customizations. In the file `sample.css`, you see each element of Inktomi Search Software's user interface. Modify the element you wish to change, then save the style sheet. As always, click the **Reload** button on the **Server/Parameters** tab of the admin interface so that your changes will take effect.

## ■ Managing frame sets

A **frame set** is a user-defined HTML structure that allows you to display multiple Web pages (known as *documents*) on a single browser page.

Because a frame set is itself not a document, Inktomi Search Software treats each of the frame set's component pages as a stand-alone document. Therefore, when Inktomi Search Software locates a query term in a component document, Inktomi Search Software adds only the matching document to the results page; it does not return the entire frame set.

If you want to recreate the original frame set that contains the matching document, see the following section.

### Sending results pages to a surrogate frame set

When a user selects a search result on the results page, Inktomi Search Software opens only that document, even if it is normally displayed in a frame set. As administrator, you can have the matching document open in a frame set. Do the following:

- 1 Create a surrogate frame set in which the matching document will be displayed.
- 2 Determine whether the matching document belongs in the surrogate frame set.
- 3 Send the matching document to the surrogate frame set.

#### Creating a surrogate frame set

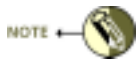
The first step is to create a page within the current document directory that acts as the surrogate frame set. This surrogate frame set duplicates the original frame set, including the number of windows and the documents displayed.

To create a surrogate frame set page, copy and edit the HTML that created the original frame set page. The surrogate frame set must go in the Inktomi Search Software `/docs` directory. Suppose the original frame set source looks like this:

```
<frameset cols="200,*" border=0>
<frame src="http://www.widget.com/activities/mainmenu.html" name="menu">
<frame src="http://www.widget.com/activities/top-homepage.html" name="display">
</frameset>
```

The code to create the surrogate frame set page would be similar to the following. The example code is in bold lettering.

```
<!--$
#page to present 'widget' frame set content pages
if query.has_key("url"): url = query["url"][0]
else: url = ""
-->
<frameset cols="200,*" border=0>
<frame src="http://www.widget.com/activities/mainmenu.html" name="menu">
<frame src="&$url;" name="display">
</frameset>
```



The “&\$url;” expression converts the `url` variable from Python to HTML.

### Determine which documents to open in the frame set

To successfully use surrogate frame sets for displaying matching documents, you need to determine the following:

- Which matching documents were originally displayed in a frame set.
- Which frame set originally contained the matching document.

In most cases, the original frame set can be identified by some feature of the URL, such as the directory in which a page resides.

### Send matching documents to the frame set

Individual search result hits are expressed by a file named `onehit1.html`, found in the `/docs` folder of the Inktomi Search Software installation directory. To create a surrogate frame set page and display matching documents in it, you must edit the `onehit1.html` file.

Locate the following code at the top of the `onehit1.html` file:

```
<!--$
if not title: title = brkurl(url,60)

width1 = 40*score/100
width2 = 40-width1
fshref = fsprfx+'&fs='+urlquote(url)+'&qt=&q1='+q1+'&st=1&nh='+str(nh)+'&lk='+lk
-->
```

To open matching documents in the surrogate frame set, add code similar to the following. The example code is in bold lettering.

```
<!--$
if not title: title = brkurl(url,60)

width1 = 40*score/100
width2 = 40-width1
fshref = fsprfx+'&fs='+urlquote(url)+'&qt=&q1='+q1+'&st=1&nh='+str(nh)+'&lk='+lk

origurl = url

teststring = 'http://www.widget.com/activities/'
if (url[:len(teststring)] == teststring):
    if (url[-8:] != 'notme.html'):
        url = 'http://[Inktomi Search host:port]/frameset.html?url=' + url
-->
```

The first new line:

```
origurl = url
```

creates a copy of the matching document’s URL. This new URL maps to a directory on your local server, rather than to the server on which the original document is located. The new URL can be displayed below the document summary in the search results list.

The subsequent new lines:

```
teststring = 'http://www.widget.com/activities/'
if (url[:len(teststring)] == teststring):
    if (url[-8:] != 'notme.html'):
        url = 'http://[Inktomi Search host:port]/frameset.html?url=' + url
```

start by specifying a string pattern for determining which matching documents should be opened in a frame set. In this example, the Python code compares the `teststring` string variable to the matching document's URL to determine if you need to create a special frame set page.

The generic Python expression `url[:someinteger]` (where `someinteger` is an integer or integer variable) returns the first `someinteger` characters of the `url` string variable. Therefore, the Python expression `len(teststring)` returns an integer that represents the length of the `teststring` string variable. The Python expression `url[:len(teststring)]` combines these concepts, returning the first characters of the `url` string variable, given the length of `teststring`. For example, if `len(teststring)` is 48 characters, `url[:len(teststring)]` returns the first 48 characters of the `url` string variable. The full expression `(url[:len(teststring)] == teststring)` determines whether the first `len(teststring)` characters of `url` match `teststring` exactly.

The `if (url[-8:] != 'notme.html'):` line handles a potential exception document (in this example, `notme.html`) that in this case complies with the search pattern. This file name would likely be that of the original frame set. The `url[-8:]` (note the negative and the position of the colon), means “the last 8 characters of the `url` string variable.” Note that the number specified in the code changes depending on the length of the excepted file name.

The last line creates a replacement URL by assembling the following:

- The URL for the surrogate page, followed by
- The expression `?url=` , followed by
- The original URL

Finally, change this line:

```
&$brkurl(url,60);
```

to:

```
&$brkurl(origurl,60);
```

This causes the matching document's original URL to be displayed below the document summary in the search results list.

## Changing the `patches.py` file



*Products*

This section contains general information about changing the `patches.py` file:

- [About Python and `patches.py`](#), on page **88**
- [Monitoring changes](#), on page **91**
- [Default values for new collections](#), on page **92**

## ■ About Python and `patches.py`

For advanced customization of Inktomi Search Software, change `patches.py`. This file is written in the Python programming language. Find out more information about Python by visiting their Web site at: <http://www.python.org>

There you'll also find an online tutorial. Books on Python include *Internet Programming with Python* from M&T Books and *Programming Python* from O'Reilly & Associates, Inc.

Python code uses indentation to separate program blocks, just like C code uses { } (curly braces). When editing any of the Python in `patches.py`, it is **extremely important** that the original indentation of the file not be corrupted. This can break the Python embedded pages, making them return an error rather than executing.

The `patches.py` file provides a solid structure that allows you to customize Inktomi Search Software while preserving its current functionality. Your changes act as a “wrapper” or a “patch” around the old functions.

The `patches.py` file is thoroughly commented. You should read the comments for extended documentation on the `patches.py` file. The file is made up of several skeletal functions that provide the groundwork for your changes. Each skeleton has a similar structure, containing an “old” and a “new” procedure. Initially, the “new” procedure is declared as being associated with the “old” procedure. The “old” procedure is then called and changed according to your specifications. After it is changed, the procedure is renamed to the “new” procedure. Finally, the “old” procedure is “smashed” and the “new” procedure is introduced. A general guideline is to add code to the “new” procedure that modifies parameters before the call to the “old” procedure.

Below are skeletons in the `patches.py` file. Notice the similarity in structure of each skeleton.

```
=====
old_getdoc = getdoc.getdoc

def new_getdoc(col,site,url,path,dict,force):
    code,msg,hdrs,doc = old_getdoc(col,site,url,path,dict,force)
    return code,msg,hdrs,doc

getdoc.getdoc = new_getdoc
=====
old_parse = parse.parse

def new_parse(col,site,doc,url,size,date,dict,doctype,params):
    (title,description,publisher,url,size,date,flags,extra,
     httpequivs,robots,fields,terms,hrefs,imgs) = old_parse(
        col,site,doc,url,size,date,dict,doctype,params)
    return (title,description,publisher,url,size,date,flags,extra,
           httpequivs,robots,fields,terms,hrefs,imgs)

parse.parse = new_parse

=====
old_quality = parse.quality
```

```

def new_quality
(col,site,doc,title,description,publisher,url,size,modtime,
idxtime,flags,nlinks,extra,httpequivs,robots,fields,terms,
hrefs,imgs,dict,doctype,params):
    quality = old_quality(
col,site,doc,title,description,publisher,url,size,modtime,idxtime,
flags,nlinks,extra,httpequivs,robots,fields,terms,hrefs,imgs,dict,
doctype,params)
    return quality

parse.quality = new_quality

=====
old_determine_language = indexer.determine_language

def new_determine_language(col,title,description,publisher,url,extra,
fields,terms,dict):
    stemlang,filtlang = old_determine_language(
col,title,description,publisher,url,extra,
fields,terms,dict)
    return stemlang,filtlang

indexer.determine_language = new_determine_language

=====
old_delete_doc = indexer.delete_doc

def new_delete_doc(col,url,dict,msg,deldocdb,deldocid):
    old_delete_doc(col,url,dict,msg,deldocdb,deldocid)

indexer.delete_doc = new_delete_doc
=====
old_delete_previous_doc = indexer.delete_previous_doc

def new_delete_previous_doc(col,url,dict,msg,prevdocdb,prevdocid):
    old_delete_previous_doc(col,url,dict,msg,prevdocdb,prevdocid)

indexer.delete_previous_doc = new_delete_previous_doc

=====
old_delete_duplicate_doc = indexer.delete_duplicate_doc

def new_delete_duplicate_doc
(col,url,dict,msg,dupurl,dupdocdb,dupdocid):
    old_delete_duplicate_doc

```

```

(col,url,dict,msg,dupurl,dupdocdb,dupdocid)

indexer.delete_duplicate_doc = new_delete_duplicate_doc
=====
old_insert = indexer.insert

def new_insert
(col,statusproc,title,description,publisher,url,size,modtime,

idxtime,flags,nlinks,extra,checksum,fields,terms,quality,
    stemlang,filtlang,dict):
    old_insert(

col,statusproc,title,description,publisher,url,size,modtime,idxtime,

flags,nlinks,extra,checksum,fields,terms,quality,stemlang,filtlang,
    dict)

indexer.insert = new_insert
=====
old_revisit_interval = spider.revisit_interval

def new_revisit_interval(col,site,url,dict):
    interval = old_revisit_interval(col,site,url,dict)
    return interval

spider.revisit_interval = new_revisit_interval

=====

```

It's helpful to use print statements to monitor your changes (see [Monitoring changes](#), on page 91). Restart the server after you edit the `patches.py` file, so your changes take effect.

## ■ Monitoring changes

Adding print statements to the `patches.py` file helps you monitor your changes. Following is an example of a print statement added to a snippet of a `patches.py` file.

```
# Log statements are added for monitoring.

import log

def new_parse(col,site,doc,url,size,date,dict,doctype,params):
    (title,description,publisher,url,size,date,flags,extra,
     httpequivs,robots,fields,terms,hrefs,imgs) =old_parse(
        col,site,doc,url,size,date,dict,doctype,params)
    log.log(log.info,"URL is " + repr(url))
    return (title,description,publisher,url,size,date,flags,extra,
            httpequivs,robots,fields,terms,hrefs,imgs)
```



**NOTE** The log levels are `log.info`, `log.warning`, and `log.error` (fatal).

The portion “URL is ” + `repr(url)`” can be any statement reflecting the type of change that has been made. After changing the `patches.py` file, restart the server. You can monitor changes by viewing the log file.

## ■ Default values for new collections

Inktomi Search Software's default collection settings should provide optimal performance when establishing new collections. However, default collection settings can easily be changed. If you find that you are using the same settings repeatedly when creating new collections, you may choose to change the default values. Default values for new collections can be modified in the `patches.py` file.

For example, setting the default user agent could be done this way:

```
import config
config.newcol_user_agent_alist = [('*', 'MyAgent')]
```

More than 50 collection parameters can be set.

**TABLE 1. New collection default values:**

<code>newcol_allow_non_western = 0</code>
<code>newcol_alt_max_raw_freq = 2</code>
<code>newcol_alt_weight = 1</code>
<code>newcol_authorization_alist = []</code>
<code>newcol_body_max_raw_freq = 8</code>
<code>newcol_body_weight = 1</code>
<code>newcol_default_ageofdoc = 60*60*24*16</code>
<code>newcol_dedup_by = 0</code>
<code>newcol_description_max_raw_freq = 2</code>
<code>newcol_description_weight = 4</code>
<code>newcol_dfltname = 'index.html'</code>
<code>newcol_disallow_cgiscrpts = 1</code>
<code>newcol_disallow_querystrings = 1</code>
<code>newcol_discover_sites = 0</code>
<code>newcol_dup_filter_alist = []</code>
<code>newcol_from_alist = []</code>
<code>newcol_filter_domino = 0</code>
<code>newcol_group_filter_alist = []</code>
<code>newcol_http_header_alist = []</code>
<code>newcol_iii_maxdocs = 200</code>
<code>newcol_imaxmsgs = 25</code>
<code>newcol_incremental_merge_min = 5000</code>
<code>newcol_incremental_merge_ratio = 8.0</code>
<code>newcol_keep_alives = 1</code>
<code>newcol_keyword_max_raw_freq = 2</code>
<code>newcol_keyword_weight = 4</code>
<code>newcol_maxbucketsecs = 60*60*24*32</code>

TABLE 1. New collection default values:

<code>newcol_maxdirdepth = 10</code>
<code>newcol_maxhops = 25</code>
<code>newcol_maxhops_alist = []</code>
<code>newcol_maxmsgs = 500</code>
<code>newcol_maxrecvlen = 1&lt;&lt;20</code>
<code>newcol_maxthreads = 5</code>
<code>newcol_minbucketsecs = 60*60*24</code>
<code>newcol_mirror_poll_interval = 60*60</code>
<code>newcol_netnews_poll_interval = 60*10</code>
<code>newcol_numbuckets = 6</code>
<code>newcol_numstrikes = 3</code>
<code>newcol_parse_cpulimit = 60</code>
<code>newcol_preference_alist = [</code> <code>  ('http://www.*',2),</code> <code>  ('http://*',1),</code> <code>  ('*',0)]</code>
<code>newcol_propagate_hops_fast = 0</code>
<code>newcol_proxy_alist = []</code>
<code>newcol_quality_alist = [</code> <code>  ('*://*/*/*/',0),</code> <code>  ('*://*/*/',1),</code> <code>  ('*://*/',2),</code> <code>  ('*://*/*/*/',-2),</code> <code>  ('*://*/*',-1)]</code>
<code>newcol_quality_nlinks_divisor = 3</code>
<code>newcol_reindex_interval = 60*60*24*90</code>
<code>newcol_retrylatercodes = [502,503,504]</code>
<code>newcol_retrylatertime = 60*10</code>
<code>newcol_robots_override_alist = [('*',0)]</code>
<code>newcol_root_urls = []</code>
<code>newcol_score_incr = 4</code>
<code>newcol_seltime = 60</code>
<code>newcol_sleep_alist = []</code>
<code>newcol_title_max_raw_freq = 2</code>
<code>newcol_title_weight = 8</code>
<code>newcol_url_filter_alist = []</code>
<code>newcol_use_primary_name = 0</code>
<code>newcol_use_sitelist = 0</code>
<code>newcol_user_agent_alist = [('*', 'Ultraseek')]</code>
<code>newcol_scan_interval = 60*60*24</code>



## Using Content Assistants



*Products*

Inktomi Search Software lets you create a content assistant to assign topics to indexed documents, filter unwanted or inappropriate documents, or add meta-data to indexed documents.

This chapter contains the following information:

- [Understanding Content Assistants](#), on page **96**
- [Evaluating Documents](#), on page **96**
- [Example Uses](#), on page **96**
- [Understanding the Service Provider Interface](#), on page **97**

## ■ Understanding Content Assistants

A Content Assistant Service is a program that you create to interact with the Inktomi Search Software (ISS) to define and assign topics to indexed documents, filter unwanted or inappropriate documents, or add meta-data to indexed documents. The *Inktomi Search Content Classification Engine Guide* provides a complete description of the interface between the ISS and one or more Content Assistant Services that you create and also provides a sample content assistant implemented in Java.



**IMPORTANT** A special license key is required in order to access Content Assistant functionality. You must also have a CCE license if you plan to push topics to the Inktomi Search Software from a content assistant.



**NOTE** You cannot use a content assistant to inject new documents into the Inktomi Search Index.

## Evaluating Documents

When the ISS begins indexing documents, it will send each document to each of your content services for evaluation. Each content service can use its own criteria to evaluate each document. The result of each document evaluation always results in one of the following outcomes:

- The content assistant service takes no action on the document and the ISS indexing of that document is not affected.
- The ISS is instructed by the content assistant to filter the document, due to inappropriate, unwanted, or irrelevant document contents.
- Any topics and meta-data assigned to the document by a content assistant service will be stored by the ISS.

## Example Uses

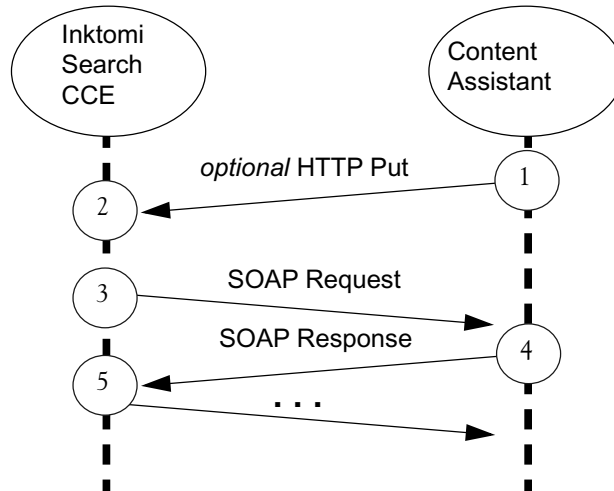
- Specify topic information for a document, which is stored by the ISS.
- Identify documents containing obscene or pornographic material that are to be filtered by the ISS.
- Add meta-data containing information such as reviews or ratings to a document, based on the document's content.
- Replace the title and description information for an indexed document.

## ■ Understanding the Service Provider Interface

Your content service and the ISS communicate using a sequence of transactions, as shown in [Figure 41](#). For a complete description of the interface between the ISS and one or more Content Assistant Services that you create, refer to the *Inktomi Search Content Classification Engine Guide*.

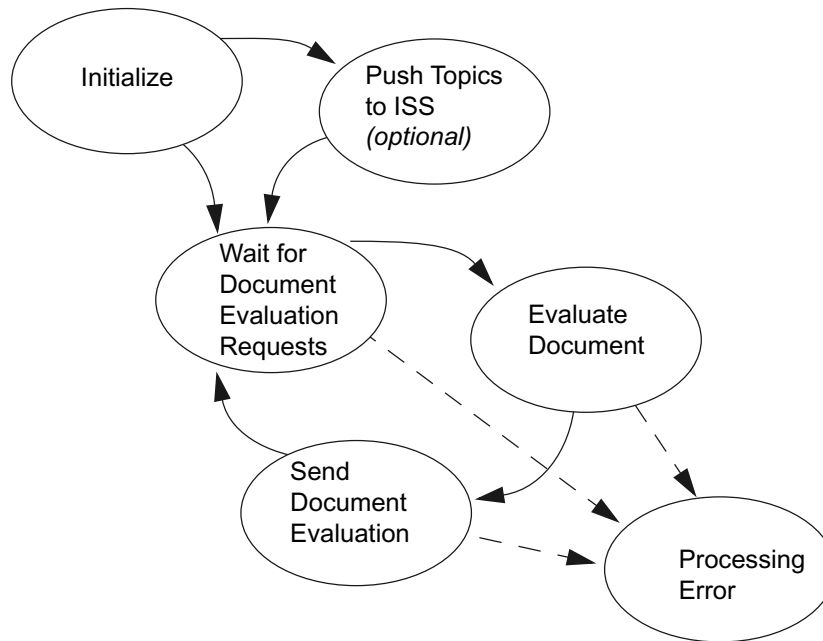
- 1 Your content assistant may optionally choose to transmit a document topic hierarchy to the ISS using HTTP PUT.
- 2 The ISS stores or updates the topic hierarchy. Any new topics are merged with the existing hierarchy. Topic information that already exists in the hierarchy, will be replaced by the new information.
- 3 When the ISS begins re-indexing its documents, it will invoke a SOAP procedure to ask your content assistant to evaluate each document. You pre-configure the ISS to send the type of information your content assistant needs to evaluate each document. The document re-indexing process is controlled and configured using the ISS admin interface.
- 4 Your content assistant processes the SOAP request, evaluates the document using criteria you define, and then invokes another SOAP procedure to send the results of the document evaluation back to the ISS.
- 5 The ISS processes the response and update the information for the document. This process continues until all documents all have been evaluated or until an error occurs.

FIGURE 41 Transactions between Content Assistant and ISS.



## Content Assistant States

FIGURE 42 Content Assistant State Transitions



## Customization Examples and Templates



*Products*

This section contains specific examples, templates, and code snippets that you can use to customize Inktomi Search Software:

- [Customizing the user interface](#), on page **100**
- [Customizing index behavior](#), on page **110**

## ■ Customizing the user interface

This section contains information about customizing the Inktomi Search Software interface. Most modifications can be made through editing the HTML of various documents in the `/docs` directory. However, some simple user interface customizations can be achieved through the admin interface. See [Customizing with the Admin Interface](#), on page 11, for more information.

Before you edit the files in the `/docs` directory, read [Getting ready to customize](#), on page 47.

### Adding a logo to every page

To place your organization's logo or navigation bar at the top of every page, you add a single line of HTML to the bottom of the `header.html` file. Following is an example of the `header.html` file with an "image" tag added to the end (changes are in bold text):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 final//EN">
<html>
<head>
<title>&$config.hostname:: &$title;</title>
</head>
<body bgcolor="#ffffff" link="#0000ee" vlink="#551a8b">

```

To center the logo, add the "center" container tags:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 final//EN">
<html>
<head>
<title>&$config.hostname:: &$title;</title>
</head>

<body bgcolor="#ffffff" link="#0000ee" vlink="#551a8b">
<center>

</center>
```

To add your logo to the bottom of every page, insert the same HTML tags in the `footer.html` file. Add the tags to the top of the file:

```
<center>

</center>
</body>
</html>
```

To reload the pages through the Web server go to the **Server/Parameters** tab in the admin interface and click **RELOAD**.

### Adding a logo to the search box

The search page is generated by the `queryform_*.html` series of files. (For more information, see [The queryform\\_\\*.html series](#), on page 31.)

To position your logo above the search box, insert a single line of HTML in the search form. Look for the appropriate table “td” tag and place your “image” tag below it:

```
<div class=query>
<table cellspacing=0 cellpadding=6 border=1 width="100%">
<tr valign=baseline>
<td bgcolor="#ffffff" class=query>

<p><b class=label>Search:</b>
```

To reload the pages through the Web server go to the **Server/Parameters** tab in the admin interface and click **RELOAD**.

## Adding a search box to your home page

Putting a search box on your home page makes it easy for users to immediately find what they're looking for on your site. In this section, you create a search form that mimics the table structure of Inktomi Search Software's default search interface. However, it is also possible to create a stand-alone search page, where you build the form from scratch. To learn more, see [Creating a stand-alone search form](#), on page 101.

To add a search box, go to the search page on the port you have designated and choose “view source” from your browser. Look for the following “form” container tag:

```
<form name=seek1 method=GET action="/query.html">
```

with its companion:

```
</form>
```

Highlight everything from the opening “form” tag to the closing “form” tag, then copy and paste it to the new location on your home page. Then, edit the existing URLs in this text to match the location you are using for your search engine. For example, change:

```
<form action="/query.html" method="GET">
```

to read:

```
<form action="http://[search host]/query.html" method="GET">
```

And change:

```

```

to read:

```

```

To reload the pages through the Web server go to **Server Parameters** in the admin interface and click **Reload**.

## Creating a stand-alone search form

It is possible to create an original, custom search page that sends queries to `query.html`. Since such a page would not be built through the `indexform_.html` series (see [The indexform\\_.html series](#), on page 27), you have the flexibility to design a unique user interface from scratch.

To accomplish this, build the HTML just like you normally would for other pages in your Web site.

Decide where the search box should go, then add the following form elements:

```
<form action="http://[search server host:port]/query.html" method="GET">
<input type="text" size="30" maxlength="35" name="qt"><br>
<input type="submit" value=" Seek ">
</form>
```

On your custom page, the above will display a simple text box and a submit button labeled “seek”. Add instructions such as “Search Widget Company” by simply adding the HTML text above the search box. Links to Help or Advanced search can be added by adding the following:

```
<a href="http://[search server host:port]/help/">Help</a>&nbsp;&nbsp;&nbsp;  
<a href="http://[search server host:port]?q=a">Advanced</a>
```

Be sure to add parameters, such as “qp” or “qc” where necessary. For more information about these parameters, refer to [Changing the results page](#), on [page 52](#).

Find out more about displaying different collections for different interfaces in the next section.

You may also wish to copy other elements such as rotating tips, from the existing search form. The simplest way to do this is to open a browser and point it to the existing search page. Do a “view source” in the browser, and cut and paste the appropriate snippet into your custom page.

## Limiting the collections displayed in the search box

The collections you create in Inktomi Search Software display inside the search box. However, you may not want to display and search all of your collections. You can change Inktomi Search Software so that a search box on one page displays and searches certain collections, and a search box on another page displays and searches other collections. This can be beneficial if you have documents that reside in classified areas, where only certain individuals are authorized to search. The easiest way to accomplish this is to clear the checkboxes labeled “Search this collection by default” and “Show this collection by default” under the **Collections/Tuning** tab in the admin interface. You can also do it with the form variables shown below.

## qc, col, and qp alternatives

If the initial search page is a custom page with a stand-alone search form (see [Creating a stand-alone search form, on page 101](#)), there is a simple alternative. Simply set the “qc” and “col” variables to control which collections are displayed and searched.

```
<form action="http://[search server host:port]/query.html" method="GET">
<input type="text" size="30" maxlength="35" name="qt"><br>
<input type="submit" value=" Seek ">
<input type="hidden" name="col" value="widget">
<input type="hidden" name="qc" value="widget">
</form>
```

See [qc](#), on page 69 and [col](#), on page 53 for more information on those parameters.

Another alternative for displaying some, but not all collections is to not display “real” collections at all. The “qp” form variable allows you to create “virtual collections”, and gives you the flexibility to display whatever you want. See [qp](#), on [page 60](#) for more information.

## Limiting the collections displayed in the default search interface

Sometimes it is necessary, though, to change the values in the `indexform_.html` and `queryform_.html` series.

For example, suppose you have several collections, entitled, “apples”, “oranges”, “red”, and “green”. You want to enable users to search the “apples” and “oranges” collections, but not “red” or “green”. To do that, open `query.html` and `index.html`. In each file, change the line that sets the default value for the “qc” (query collections) variable. The “qc” variable specifies which collections display on the search form. To display only the “apples” and “oranges” collections, change the “else” line:

```
if query.has_key("qc"): qc = string.join(query["qc"])
else: qc = ""
if query.has_key("rf"): rf = string.atoi(query["rf"][0])
else: rf = config.default_rf
```

to this (changes are in bold):

```
if query.has_key("qc"): qc = string.join(query["qc"])
else: qc = "apples oranges"
if query.has_key("rf"): rf = string.atoi(query["rf"][0])
else: rf = config.default_rf
```

Reload the cached pages so that your changes take effect. To reload the pages through the Web server, go to **Server Parameters** in the admin interface and click **Reload**. As a result, only the “apples” and “oranges” collections display in the search box. Be sure to use the collection’s internal name (as seen on the bottom of the Tuning parameters of Collections in the admin interface), not the display name.



Both `index.html` and `query.html` must be modified in order to limit the collections displayed in both the initial search pages and the search results pages.

Now, remember that “qc” only determines which collections to *display*. You also want to consider which collections to *search*. If you also want to limit which collections to search, you follow the same procedure, only modify the settings for the “col” variable. So again in `index.html` and `query.html`, change this:

```
if query.has_key("col"): col = string.join(query["col"])
else: col = ""
if query.has_key("nh"): nh = string.atoi(query["nh"][0])
else: nh = config.default_nh
```

to this (changes are in bold):

```
if query.has_key("col"): col = string.join(query["col"])
else: col = "apples oranges"
if query.has_key("nh"): nh = string.atoi(query["nh"][0])
else: nh = config.default_nh
```

To reload the pages through the Web server, go to **Server Parameters** in the admin interface and click **Reload**.

## Displaying each collection name on a separate line

In the search box, each collection name with its corresponding checkbox is displayed next to each other, wrapping to the next line when necessary. A common modification is to put each collection name on a separate line.

You can accomplish this by modifying `indexform.html`, `indexforma.html`, `queryform0.html`, and `queryform0a.html`. It is important to modify each of these files, so that the display is consistent throughout the interface.

In `indexform.html` and `indexforma.html`, look for the following block (does not fit on page):



The **[same line]** indicator in the snippets means that line of code should be on the same line as the line above it. We are unable to properly display the code due to formatting restrictions. Spacing is important. The following example contains only six lines of code.

```
<!--$
    for c in qcols:
        --><nobr><!--$
            if not c in cols: --><input type=checkbox name=col value="&$c;"
[same line]onclick="document.&$formname;.rq[0].checked=true"><!--$
                else: --><input type=checkbox name=col value="&$c;" checked
[same line]onclick="document.&$formname;.rq[0].checked=true"><!--$
                    write(string.replace(htmlquote(self.text(c.pretty_name)),
[same line]u' ', '&nbsp;'))
                        --></nobr>
```

Put a break at the end of the block:

```
<!--$
    for c in qcols:
        if not c in cols: --><input type=checkbox name=col value="&$c;"
[same line]onclick="document.&$formname;.rq[0].checked=true"><!--$
            else: --><input type=checkbox name=col value="&$c;" checked
[same line]onclick="document.&$formname;.rq[0].checked=true"><!--$
                write(string.join(string.split(htmlquote(c.pretty_name)), '&#160;'))
                    --><br>
```

In the `queryform0.html` and `queryform0a.html`, look for the following block of code:

```
else:
    for c in qcols:
        --><nobr><!--$
            if not c in cols: --><input type=checkbox name=col value="&$c;"
[same line]onclick="document.&$formname;.rq[0].checked=true"><!--$
                else: --><input type=checkbox name=col value="&$c;" checked
[same line]onclick="document.&$formname;.rq[0].checked=true"><!--$
                    write(string.replace(htmlquote(self.text(c.pretty_name)), u' ', '&nbsp;'))
                        --></nobr>
```

Again, put a break at the end of the block:

```
<!--$
else:
    for c in qcols:
```

```

        if not c in cols: --><input type=checkbox name=col
[same line]value="&$c.name;"><!--$
        else: --><input type=checkbox name=col value="&$c.name;" checked><!--$
        write(string.join(string.split(htmlquote(c.pretty_name)), '&#160;'))
        --><br>

```

To reload the pages through the Web server go to **Server Parameters** in the admin interface and click **Reload**.

## Displaying two URLs per hit

Sometimes, multiple files may have the same content, but different file formats. It is possible to display each link in the same hit, allowing the user to decide which file format to use.

For example, suppose Widget company has a set of files that are available in both text and TIFF formats. To display links to both formats in the search results, look for the following block in `onehit1.html` and `onehit2.html`:

```

<b class=title><a href="&$url;"><!--$write(title);--></a></b><br>

```

This is the part of the file that shows the title of a hit, linking it to the indexed text file. Add a few more lines to the file to add a link to the corresponding TIFF file:

```

<b class=title><a href="&$url;"><!--$write(title);--></a></b><br>
<!--$
if url[-4:]==' .txt':
    tifurl = url[:-4]+' .tif'
    --><a href="&$tifurl;">TIFF</a><br>

```

The link to the TIFF file will be added, along with the label, “TIFF”.

To reload the pages through the Web server go to **Server Parameters** in the admin interface and click **Reload**.

## Adding an image to each search result

An interesting customization that can be done fairly easily is to associate a small image with each search result. For example, suppose Widget company has a media department that puts out several publications. The Widget Webmaster decides to add an image to each search result that reflects which publication is associated with that hit (all hits from *Widget World* have `world.gif`, all hits from *Widget Works* have `cover.gif`, etc.).

To accomplish this, open the file `onehit.html`. Look for the first “td” tag toward the beginning, which contains the block shown below.



NOTE

The **[same line]** indicator in the snippets means that line of code should be on the same line as the line above it. We are unable to properly display the code due to formatting restrictions. Spacing is important.

```

<td width="95%">
<!--$

```

```

if ct is not None and not qti:
    topic_dict = topic.current.urls.get(string.lower(url))
    if topic_dict is not None:
        if topic_dict.has_key(ct):
            stars = topic_dict[ct]
            for i in range(0,stars):
                --><!--$
-->
<b class=title><a href="&$url;"><!--$write(title);--></a></b><br>
<span class=summary><!--$write(summary);--></span><br>
<font size="-1" class=publisher><i>&$publisher;</i></font>
<font size="-1" class=url><i>&$brkurl(url,70); -</i></font>
<font size="-1" class=size><i>size&#160;&$%.1f"%(size/1024.0);K </i></font>
<i class=collections><!--$
first = 1
for docdb in docdb:
    if not dbcol.has_key(docdb): continue
    if not first: -->, <!--$
    else:
        first = 0
        -->- <!--$
        -->&$dbcol[docdb].pretty_name;<!--$
--></i><span class=wordscores><!--$
if ws:
    first = 1
    for term,scor in dterms:
        if not first: -->, <!--$
        else:
            first = 0
            --><br>
<!--$
        -->&$term;: &$max(int(scor+0.5),1);<!--$
-->
</span>
</td>

```

Change that block with the following (changes are in bold):

```

<td width="95%">
<btable>
<!--$
if (url[:19] == "http://world.widget"):
    --><a href="http://world.widget.com/"></a><!--$
if (url[:19] == "http://works.widget"):
    --><a href="http://works.widget.com/"></a><!--$
if (url[:13] == "http://widget"):
    --><a href="http://widget/" ></a><!--$
-->
</table>
<!--$
if ct is not None and not qti:
    topic_dict = topic.current.urls.get(string.lower(url))
    if topic_dict is not None:
        if topic_dict.has_key(ct):
            stars = topic_dict[ct]

```

```

        for i in range(0,stars):
            -->
[same line]<!--$
-->
<b class=title><a href="$url;"><!--$write(title);--></a></b><br>
<span class=summary><!--$write(summary);--></span><br>
<font size="-1" class=publisher><i>&$publisher;</i></font>
<font size="-1" class=url><i>&$brkurl(url,70); -</i></font>
<font size="-1" class=size><i>size&#160;&$"%1f"%(size/1024.0);K </i></font>
<i class=collections><!--$
first = 1
for docdb in docdb:
    if not dbcol.has_key(docdb): continue
    if not first: -->, <!--$
    else:
        first = 0
        -->- <!--$
        -->&$dbcol[docdb].pretty_name;<!--$
--></i><span class=wordscores><!--$
if ws:
    first = 1
    for term,scor in dterms:
        if not first: -->, <!--$
        else:
            first = 0
            --><br>
<!--$
        -->&$term;: &$max(int(scor+0.5),1);<!--$
-->
</span>
</td>

```

This block tells the search engine that if a document comes from anywhere in the `world.widget` site, let the image be `wrld.gif`. If the document comes from anywhere in the `works.widget` site, let the image be `cover.gif`. If the document comes from anywhere else, let the image be `widget.gif`.

Notice that the embedded table does not have a specified width. You may wish to set a table width for the image based on your settings.

Notice also the structure of each “image” tag. In the first section, the Widget World image is located in Inktomi Search Software’s images directory and is easily retrieved. In the second section, the pointer to the Widget Works image is a fully-extended URL that resides outside of Inktomi Search Software’s directory structure. The image itself is a thumbnail of the magazine cover, which changes monthly. By pointing to the image through a fully-extended URL, the search results display will always reflect the most recent month’s magazine cover thumbnail. However, remember that pointing to images that reside outside the server will slow down the process somewhat.

Also, an important point to consider is that the integer following the (`url[ : designation` *changes* according to the URL pattern you use. For example, look at this line:

```
if (url[:19] == "http://world.widget"):
```

The number following the URL designation is 19. That is because the URL prefix that follows,

```
http://world.widget
```

contains 19 characters. In other words, the code is telling Inktomi Search Software to look for the first 19 characters in the URL, and if it matches the pattern `http://world.widget`, then display the `world.gif` image. If you had changed the URL prefix to `http://world.widget.com`, the integer would have to change to 23.

The outcome is a results page where each hit also has an image from either Widget World, Widget Works, or just plain Widget.

## Changing other content of the results page

Standard items that appear on the results page are:

- document title (string)
- summary (string)
- URL (string)
- size (integer)
- relevance ranking (integer)
- document date (integer)

For each document it indexes, Inktomi Search Software stores the following additional values:

- publisher (string)
- site (string)
- link (string)

To add or subtract from the list of items that Inktomi Search stores, change the values in the Title Record pane on the Collections/Tuning tab in the admin interface.

## Creating a non-tables version of public pages

The architecture of the results pages relies heavily on tables, so it is recommended that Inktomi Search Software is used through a browser that supports tables. However, because Inktomi Search Software is fully customizable, you can develop a non-tables version of the search and help interfaces.

To work around tables on the search form page, create an alternate search form page. The new form doesn't have to be on Inktomi Search Software's built-in Web server. Just set the "Home URL" parameter on the "parameter" card of the **Server Administration** view to point to your new form.

To work around tables in the help interface, change the `header.html` and `footer.html` files in the `/help` directory. The common graphic elements on the help pages are concentrated in the header and footer, so you can completely change the look of the help pages by changing these two files. For more information about changing headers and footers, refer to [Help header and footer files](#), on page 43.

To work around tables on the results pages, create an alternate results page. To do this, you create an HTML file that mimics the nature of the `query.html` file. The HTML file you create processes queries and displays results in a different way. By changing the settings of the form variables, you further change the way results are displayed, as described in the next section.

## Changing the rotating tips file

An easy way to customize Inktomi Search Software for your organization is to change the tips file. Some administrators have implemented tips that use product names, industry terms, and executives' proper names when changing the tips file.

To change a tip file, find the appropriate language-encoded file (en.xml is for English).

For Unix and Linux installations, look in the directory /opt/languages/.

For Windows installations, look in the directory c:\Program Files\Inktomi\InktomiSearch4.1\languages\ directory in Windows NT.

The tip file can be edited with any text editor. Before editing, make a backup copy of the file in case you make an error or wish to revert to the original file.

Add your tip by typing the entire tip on one line and pressing return at the end of the line. Each tip must occupy a separate line because they are randomly grabbed according to line. To accentuate items within the tip line, add HTML tags.

For example:

```
<tip><b class=label>Tip:</b> Capitalize proper names.<p><b class=label>Example:</b> Bill Gates</tip>
```

After you complete the new tip file, point Inktomi Search Software to the new file location. Go to the **Server Administration** view, find “Tip file” on the parameters card, and type in the new location.

## ■ Customizing index behavior

### Adding URLs to the index

Inktomi Search Software automatically adds URLs to its index if they are allowed by the filters and meet one of the following criteria:

- There is a link to them from a document the spider is crawling in the form of an “href”, “frame src”, or image map
- There is an entry in the `sitelist.txt` file for that server or URL
- You add the file using the “Add URL” command manually or automatically. For information on adding a URL automatically, refer to [Adding URLs automatically](#), on page 110.

Get more detailed information on `sitelist.txt` usage at:

<http://www.inktomi.com/products/search/support/docs/sitelist.html>

The easiest way to add URLs is through links. A simple way to add links is to create a page that acts as a directory of your files, with each filename acting as a link. Then, add the directory filename to the **Collection/Roots** tab in the admin interface.

### Adding URLs automatically

You can add URLs manually through “add URL” in the Help section. This is an easy way to add a URL from any location. You can also use the sample Java scripts provided with the XPA API to add URLs. For more information, see the *Inktomi XPA Search API Programmer’s Guide*.

### Changing the date sort

One of Inktomi Search Software’s best features is its ability to sort search results by date. The date Inktomi Search Software looks for is the last modified date of the document. You can encode the date in a Meta tag and our search engine will use that date as the date of the document. This is the preferred method for many sites. We recommend that you use the W3C profile of the ISO 8601 format for your date fields:

```
<meta name="date" content="1999-09-09">
```

The W3C profile of the ISO 8601 format is outlined here:

<http://www.w3.org/TR/NOTE-datetime>

### Changing document titles

The document title is a prominent part of search results. Inktomi Search Software extracts the information contained in the HTML “title” tag to create a title for the results page. To change the document’s title, update the information contained in the “title” tags:

```
<head>
<title>Edgar Allen Poe</title>
</head>
```

The title can also be extracted from a custom Meta field. To do this, go to the HTML Meta Tag Names

section of the **Collections/Tuning** tab in the admin interface. Set the Meta field name in the **Description:** box.

## Changing document summaries

Document summaries provide a general indication of the type of information in a document. If a document does not have a summary, Inktomi Search Software creates one by extracting relevant information from the beginning of the document. You can create a summary to provide a better guide to the type of information a document contains.

The best way to create a document summary is to place an HTML “meta” tag within the “head” container tag of the document to be indexed:

```
<meta name="description" content="Check out our collection of Frequently Asked Questions to learn more about Inktomi Search Software. Use Inktomi Search Software to search through the FAQ.">
```

The summary can also be extracted from a custom Meta field. To do this, go to the HTML Meta Tag Names section of the **Collections/Tuning** tab in the admin interface. Set the Meta field name in the **Description:** box.

Inktomi Search Software will not recognize the new summary until the document is revisited.

## Returning ALL results for a particular query

When Inktomi Search Software returns search results, it only shows the first 500 hits for a particular query, even though it may have actually found considerably more hits than that.

If you need to get a list of all the search results for a particular query, you use the file `pyqueryall.spy` to create the list. The file `pyqueryall.spy` exists expressly for this purpose.

You will find `pyqueryall.spy` in the `/docs` directory of your Inktomi Search Software installation.

Simply create a search form that points to `pyqueryall.spy` instead of `query.html`. Below is a simple search form that calls `pyqueryall.spy`:

```
<form name="seek1" method=GET action="http://www.widget.com:8765/
[same line]pyqueryall.spy">
<p><b>Search:</b></p>
<input type=checkbox name=col value="widget" checked
[same line]onclick="document.seek1.rq[0].checked=true">Widget&#160;Division<br>
<input type=checkbox name=col value="gadget" checked
[same line]onclick="document.seek1.rq[0].checked=true">Gget&#160;Division<br>
<input type=text name=qt size=40 value="" maxlength=2033>
<input type=submit value=" seek ">
</form>
```

FIGURE 43 Specifying a custom query form.

**Search:**

☒ Widget Division

☒ Gget Division

If you create an HTML page that includes the above form and submit a search on an uncommon term so that you can see the results in a browser. At the bottom of the results list, you will see a number that reflects the total number of hits for that query. It will look something like this:

```
'ndocs': 83}
```

In the above example, the number of documents for that query is 83.

When you employ `pyqueryall.spy`, remember that there is a reason why Inktomi Search Software truncates search results after the 500th hit! A very large set of results may wreak havoc on your browser. The purpose of the `pyqueryall.spy` file is to give you a way to easily parse complete lists of search results. It should not be used as a substitute to the search user interface.

## Indexing multiple fields for a single field search

It is quite common for large Web sites to have multiple authors. Sometimes it is difficult to get those authors to employ the same standards when they publish documents to the Web site. A common problem is having inconsistent Meta tags throughout the Web site.

If you have a situation where you need Inktomi Search Software to recognize different Meta tags as the same, you can modify the “patch to parse” routine in `patches.py`. Consider the following example:

Various Web authors have chosen to use either an “author” or “docauthor” Meta tag in their documents. In their search forms, they would like to perform an `author:` field search. But by default, they would miss those documents which employ a “docauthor” Meta tag.

To solve this, the Web administrator employs a patch that modifies the “patch to parse” routine in `patches.py`. The patch tells Inktomi Search Software to look for “docauthor” Meta tags in documents and insert a corresponding “author” field association. So “author” searches work for both “author” and “docauthor” documents. (`author:` would become the standard for the field search).

The Web administrator first backs up her `patches.py` file, then she looks for the “patch to parse” routine in `patches.py`:

```
old_parse = parse.parse
```

```
def new_parse(col,site,doc,url,size,date,dict,doctype,params):
    (title,description,publisher,url,size,date,flags,extra,
     httpequivs,robots,fields,terms,hrefs,imgs) = old_parse(
        col,site,doc,url,size,date,dict,doctype,params)
    return (title,description,publisher,url,size,date,flags,extra,
           httpequivs,robots,fields,terms,hrefs,imgs)
```

```
parse.parse = new_parse
```

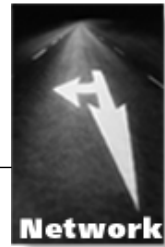
Finally the administrator changes the routine as follows (changes are in bold):

```
def new_parse(col,site,doc,url,size,date,dict,doctype,params):
    (title,description,publisher,url,size,date,flags,extra,
     httpequivs,robots,fields,terms,hrefs,imgs) = old_parse(
        col,site,doc,url,size,date,dict,doctype,params)
    for text,fieldname,weight,stop,stem,incr,max_raw_freq in fields:
    # look for a "docauthor:" field...
        if fieldname == "docauthor:":
        # insert the corresponding "author:" field
            fields.append ((text,'author:',weight,stop,stem,incr,max_raw_freq))
    return (title,description,publisher,url,size,date,flags,extra,
           httpequivs,robots,fields,terms,hrefs,imgs)
old_parse = parse.parse
```

Use print statements to monitor your changes (refer to [Monitoring changes](#), on page 91). Restart the server after you edit the `patches.py` file, so your changes take effect.



## Search Syntax



*Products*

This appendix describes the Inktomi Enterprise Search syntax and contains the following sections:

- [Why proper search syntax is important](#), on page **116**
- [Phrases, proper names, and capitalization](#), on page **117**
- [Using plus and minus operators](#), on page **118**
- [Double-pipe](#), on page **125**

## ■ Why proper search syntax is important

Knowing how to create refined, efficient searches is important for both users and administrators. Leveraging Inktomi Search Software's search syntax to its full potential yields efficient, specific results. Users are helped along with this through rotating tips, the help interface, and the Advanced Search form.

## ■ Phrases, proper names, and capitalization

It is important to identify phrases and proper names when constructing a query. In each case, Inktomi Search Software is instructed to match words in consecutive order.

### Phrases

Consider the following query:

```
submit resume employment job opportunities
```

This query will produce better results if phrases are identified. Surround phrases with quotation marks to identify them:

```
"submit resume" employment "job opportunities"
```

In the above example, Inktomi Search Software will recognize the two phrases. It is apparent that the word `submit` must appear adjacent to the word `resume`. The same goes for the phrase `job opportunities`.

### Proper names

Proper names are interpreted in a similar way. If a proper name is identified, terms will appear next to each other. For example, consider the following query:

```
john doe
```

Resulting hits will contain the words `john` and/or `doe`, but not necessarily together. To be identified as a proper name, the name must have initial capital letters:

```
John Doe
```

Resulting hits will now contain the proper name `John Doe`.

Separate multiple proper names with a comma:

```
John Doe, James Fenimore Cooper
```

Without the comma, the entire query would be interpreted as one continuous name.

### Capitalization

If a query is submitted entirely in lower case, case is ignored in the resulting hits. However, if any capitalization is used, case is exactly matched.

For example, a query for `next` will return documents containing the words `next`, `Next`, and `NeXT`. However, a query for `NeXT` will only return documents containing the word with exactly matching case: `NeXT`.

## ■ Using plus and minus operators

The plus and minus operators are useful tools that enhance search results by allowing you to exclude or include terms.

### Plus

If a term must appear in the search results, put a plus in front of it:

```
product price +widget
```

In the above example, the words `product` and `price` might be included in the search results. However, the word `widget` *must* occur in each hit. The plus sign can be used with other search syntax tools:

```
+site:www.widgets.com +"new and improved"
```

In the above example, the plus sign is used in conjunction with the site field search. The plus sign is also used with quotation marks to require a phrase.

### Minus

Put a minus sign in front of terms that are not desired in search results:

```
bats -baseball
```

Search results will prefer documents that contain information about *bats* (flying rodents), over *bats* (baseball bats). However, the minus operator does not necessarily exclude terms altogether. Documents that contain terms marked with a minus operator will be excluded from the search results.

## Field searches

A field search is a search that looks for certain elements within a document. These elements may be in any of the following formats:

- title
- URL
- site
- link
- imagelink
- alt (image)
- description
- keywords
- "Meta" data
- Dublin Core "Meta" data
- doctype
- language
- collection
- topic

For example, suppose a user wanted to search for the phrase “New Year’s Resolution”. However, the user only wants documents returned which contain that phrase in the title. Documents containing the phrase in the body or elsewhere should be ignored. The user could construct a query that looks like this:

```
title:"new year's resolution"
```

The user can also use the Advanced Search form to create the query, in which case he need not know the exact syntax to create the search. However, knowing this syntax is extremely useful for the administrator who is populating a “qp” form variable that restricts searches to certain document titles.

This same format of `field:text` is followed for all field searches. Field searches can become completely customized by extracting “Meta” data for the field search (see “Using “Meta” searches” on page 121).

## Searches by document title

You can search for an HTML document’s title by typing all or part of a title as your search query. The format is the same as for other field searches:

```
title:year
```

or:

```
title:year title:2000
```

The first query yields a broad search; the second query yields a more narrow search. To ensure that a particular phrase is found within a title field search, enclose the phrase in quotes:

```
title:"year 2000 compliance"
```

The above example yields the most narrowed pool of search results.

## Searching by URL

You can search on any document within a given URL. The URL can include any subdirectories. This

enables you to perform a broad search:

```
url:www.inktom.com
```

or a more narrowed search:

```
url:www.inktom.com/products/search
```

You would get the same results if you typed in a more complete version of the same URL search:

```
url:http://www.inktom.com/products/search
```

These queries yield all documents within the specified directories. However, you can also search for a particular word as part of the URL. For example, the query:

```
url:faqs
```

returns the same documents as the query from the previous example, and also returns documents from other directories that contain the word “faqs”.

## Searching by site

You can search on any Web site within a collection. Unlike searching by URL, this search is limited to the site portion of a URL. For example, the query,

```
site:sun.com
```

yields results for `www.sun.com` and `java.sun.com`, but not `sun.co.uk`. Search by site to list all pages in a given site.

## Searching by link

Search for links within a document by using the link field search. This can be a useful tool for finding cross-references. For example, suppose the Widget Webmaster wanted to find how many documents from the `sales.widget.com` site were pointing to the `products.widget.com` site. This could be accomplished with a simple link field search:

```
+link:products.widget.com  
+site:sales.widget.com
```

Notice the use of the plus signs as requiring operators. This makes both components of the query required in the search results.

## Searching by image links

Image links are hyperlinks attached to images. Search for image links within a document by using the `imagelink` field search. This search behaves similarly to the link field search, but only hyperlinks to images are searched. For example, suppose you are looking for an image that is linked to a page containing the word `widget` in the URL. That image could be found using the image link search:

```
imagelink:widget
```

## Searching by “alt” (image)

You can search for an image within documents by looking for the text in the “alt” attribute of the “image”

HTML tag.

For example, suppose Widget company has a document that contains an image of their new Aerodynamic Widget. The “image” tag within the HTML looks something like this:

```

```

Inktomi Search Software will find this image for all image field searches that specify any of the text within the “alt” attribute:

```
alt:amazing  
alt:aero-dynamic  
alt:widget
```

## Searching by description or keywords

Users have the ability to search for elements within a document’s description or keywords. Both of these elements are created within the “Meta” tag of the document. For more information on the “Meta” tag, see [Using “Meta” searches](#).

The document’s description is the summary that appears in search results. Keywords are inserted to highlight certain topics within a document. Both are created by each document’s author.

Field searches by description or keywords follow the same format as other field searches:

```
description:Widget  
keywords:"press release"
```

## Using “Meta” searches

You can also use the “meta” HTML tag to customize field searches. For example, to create a collection that lists books and provides searches by author, embed the following “meta” tag into the HTML:

```
<meta name="author" content="Edgar Allen Poe">
```

To find a document, you can then search `author:poe`, `author:allen`, `author:edgar allen poe`, and so forth. You can set the name to any string.

Another use of the “meta” tag is to create keywords that help users find your document. The more keywords available for a document, the more likely users will find the document when searching. Below is an example of keywords you might list for a document containing information on Edgar Allen Poe:

```
<meta name="keywords" content="literature, literary, poems, poetry, book, biography,  
education, school, American, classic, Raven, Edgar Allan Poe, Edgar, Allan, Poe,  
Annabel, Lee, Nevermore, death, gothic, Eliza, Usher, fall, Edgar A. Poe, Edgar Allen  
Poe, Allen, black, cat, amontillado, cask, rue morgue, murders, dark">
```

## Using Dublin Core “Meta” searches

Dublin Core “Meta” searches are very similar to standard “Meta” searches, but with a slightly different format. A Dublin Core “Meta” tag search might look something like this:

```
dc.subject:finance
```

Resulting hits will contain the word “finance” in the Dublin Core subject element.

## Searching by doctype (document type)

To search on a specific document type, use the doctype field search. For example, to search only Microsoft Word documents, type the query:

```
doctype:microsoft
```

The document type is based on the MIME type of the document, depending on the configuration of your Web server. If your Web server has unique MIME type configuration, you can easily modify the interpreted MIME type through the admin interface. Go to Server Doctypes, and specify your unique doctype to a format that Inktomi Search Software recognizes, such as text.

## Language

If you have multiple languages enabled, you can perform searches based on a document's language. To perform a search for documents in a particular language, use the following syntax:

```
language:french
```

Resulting hits for the above example would be recognized by Inktomi Search Software as being in French. Acceptable languages for the “language” field search are as follows:

- English - en
- Dutch - nl
- Spanish - es
- French - fr
- German - de
- Italian - it
- Portuguese - pt
- Swedish - sv
- Danish - da
- Finnish - fi
- Norwegian - no
- Japanese - ja
- Korean - ko
- Chinese Simplified - zh\_cn
- Chinese Traditional - zh\_tw

## Searching by collection

To search on a specific collection, set the field search to the internal name of a collection (not the display name). You can find the internal name for a collection under **Collections / Tuning** of the admin interface. Scroll down to the bottom of the card, where it says “Internal name: [name]”.

So, suppose you wanted to limit your search to the “Loch Ness Monster” collection. That collection has the internal name “nessie”:

```
collection:nessie
```

This query will only return results from within the “Loch Ness Monster” collection.

Since the `collection: field` search does the same thing as the `col` form variable, do not use it in a stand-alone search form. The `collection: field` search is useful when limiting topics to a single collection in CCE.

## Searching by topic

To search on a specific topic, set the field search to the topic ID number for that topic. Note that you have topics only with the Content Classification Engine (CCE) add-on product. To find out more about CCE, visit:

<http://www.inktomi.com/products/search/support/cce/index.html>

You can find the topic ID for a topic under **Topics / Status** of the admin interface. Under “Current topics” at the bottom of the card, click the topic in question and look for the “ID:” value at the top of the “Edit topics” view.

So, suppose you wanted to limit your search to the “Scottish History” topic. You have determined that the topic ID for “Scottish History” is 1575973028. Set your field search as follows:

`topic:1575973028`

This query will only return results from within the “Scottish History” topic.

This is a useful field search when using the “qp” form element. For example, you could create a form that limits search to the “Scottish History” topic, using the following syntax:

```
<form action="http://[search server hostname and port]/query.html" method="GET">
<input type="hidden" name="qp" value="topic:1575973028">
<input type="text" size="30" maxlength="35" name="qt"><br>
<input type="submit" value=" Seek ">
</form>
```

With the above syntax in place, Inktomi Search Software CCE will limit the user’s search to the “Scottish History” topic.

## Pipe and Search these results

The pipe and “Search these results” are useful tools that allow users to create specific, refined queries. The pipe will tell the engine to include content on both sides of the pipe, where the string to the right of the pipe is a subset of the string to the left of the pipe. The pipe behavior mimics “Search these results” behavior. For example, suppose a user were to type the following query:

```
restaurant bistro
```

Resulting hits will contain the words `restaurant` and/or `bistro`. The user then clicks “Search these results” and enters the subsequent query:

```
pizza spaghetti
```

Results are now narrowed to those hits which also contain the words `pizza` and/or `spaghetti`. This process is called “query refinement”.

This same behavior can be duplicated in a single query by using the pipe:

```
restaurant bistro | pizza spaghetti
```

Resulting hits will contain the words `restaurant` and/or `bistro`, and also contain the words `pizza` and/or `spaghetti`.

Note that the pipe can be used repeatedly in the same query. For example, the above query can be refined even further:

```
restaurant bistro | pizza spaghetti | San Francisco "bay area"
```

## The difference between pipe and plus

There is often confusion between the pipe operator and plus operator. In many cases, both operators produce the same effect. For example, this query

```
+restaurant +pizza
```

and this query

```
restaurant | pizza
```

will return the same results. In each case, both the word `restaurant` and the word `pizza` must appear in the search results. However, by adding several more query terms to the mix, the difference becomes more apparent.

### Example 1:

```
+restaurant +bistro +pizza +spaghetti
```

### Example 2:

```
restaurant bistro | pizza spaghetti
```

In the first example, the search results must contain **all** of the terms listed. In the second example, the search results must contain **at least one term to the left** and **one term to the right** of the pipe.

## ■ Double-pipe

In search syntax, the double-pipe has exactly the same function as the pipe, but with one difference: relevance ranking. In searches that contain a double-pipe, only the content to the right of the double-pipe is used in relevance ranking. Content to the left of the double-pipe has no impact on relevance scores.

For example, consider the following query:

```
restaurant bistro || pizza spaghetti
```

The resulting hits will include content from both sides of the double-pipe, but only the words `pizza` and `spaghetti` will impact relevance scores. Hits containing these words will rank higher in the search results.

There is little use for the double-pipe in user queries, however administrators may wish to know this syntax in order to better understand the “qp” form variable. When an administrator sets a value for the “qp” (query prefix) form variable, that value is prefixed to the user’s query with a double-pipe. For example, suppose the administrator has set a “qp” value of “widgets gadgets” in a hidden form element:

```
<input type="hidden" name="qp" value="widgets gadgets">
```

Now suppose a user types in the following query:

```
parts accessories
```

Inktomi Search Software interprets the combination of “qp” and user query with the following syntax”

```
widgets gadgets || parts accessories
```

Resulting hits will contain the word `widgets` and/or `gadgets`, and will contain the word `parts` and/or `accessories`. Relevance scores will be ranked based on the words `parts` and `accessories`.



## Index



*Products*

## ■ A

active document directory **24**  
 advanced query box, controlling through  
   form variables **71**  
 alt  
   customizing searches by **120**  
 assistant **96**

## ■ C

collection, customizing searches by **122**  
 collections  
   limiting their display in search box **102**  
 collections, displaying **15**  
 Content Assistant  
   example uses of **96**  
   overview **96**  
   state transitions **98**  
   transactions with ISS **97**  
 customizing  
   by keywords **121**

## ■ D

date sort, changing in search results **110**  
 description  
   customizing searches by **121**  
 displaying collections **15**  
 docs directory **24**  
 doctype, customizing searches by **122**  
 document structure **24**  
 document summaries, changing in search  
   results **111**  
 document title  
   changing in search results **110**  
   customizing searches by **119**  
 document type, customizing searches by  
   **122**

## ■ F

field searches  
   by collection **122**  
   by description **121**  
   by doctype **122**  
   by document title **119**  
   by image **120**  
   by image links **120**  
   by keywords **121**  
   by language **122**  
   by link **120**  
   by meta info **121**  
   by site **120**  
   by topic **123**  
   by URL **119**  
 footer.html file **25, 108**  
 footers, changing in the interface **25**  
 form variables  
   col **53**  
   ct **67**  
   dt **73**  
   ex **68**  
   fl **72**  
   fs **65**  
   ht **67**  
   la **76**  
   lk **57**  
   nh **55**  
   op **71**  
   oq **59**  
   pw **64**  
   qc **69**  
   ql **65**  
   qm **69**  
   qp **60**  
   qp and qs **62**  
   qs **61**  
   qt **55**

- rf **58**
- rq **58**
- si **68**
- st **57**
- to control advanced query box **71**
- tx **73**
- ty **72**
- ws **64**
- form variablesf
  - to control query box **69**
- frames
  - about **84**
  - limitations to using **84**
  - setting up **84**

■ **H**

- header.html file **25, 100, 108**
- headers, changing in the interface **25**
- help pages, creating a non-tables version **108**
- hiding collections **15**
- home page, adding search box to **101**

■ **I**

- imagelink
  - customizing searches by **120**
- index
  - adding URLs to **110**
- indexform\_.html **27**
- initial search pages, building **27**
- Inktomi Search Server
  - criteria for automatically adding URLs to index **110**
  - tables use **108**
- interface, changing
  - adding a logo to every page **100**
  - adding a logo to the search box **100**
  - adding a search box to your home page **101**
  - changing footers **25**
  - changing headers **25**
  - limiting collections displayed in search box **102**

■ **K**

- keywords

- customizing searches by **121**

■ **L**

- language, customizing searches by **122**
- link

- customizing searches by **120**

- logo
  - adding to every page **100**
  - adding to the bottom of every page **100**
  - adding to the search box **100**
  - centering on the page **100**

■ **M**

- meta searches, using to customize searches **121**

- meta tag
  - using to create a document summary **111**
  - using to create keywords **121**
  - using to customize field searches **121**

■ **N**

- non-tables version, creating **108**

■ **P**

- parameters **52, 71**
- patches.py file
  - adding print statements to **91**
  - example changes **91**
  - monitoring changes in **91**
  - procedures in **88**
  - skeleton code examples **88**
- print statements, adding to patches.py file **91**
- pyqueryall.spy **111**
- Python programming language, about **88**
- Python programming language, embedded in HTML **24**

■ **Q**

- qc variable **69, 103**
- qp variable **60**
- qs variable **61**
- query box, controlling through form variables **69**

- query collection variable **69**
- query collections variable **103**
- query prefix variable **60**
- query suffix variable **61**
- query.html file **100, 103, 108**
- queryform\_.html **31**

## ■ R

- related reading **10**
- results page
  - changing through form variables **52**
  - creating a non-tables version **108**

## ■ S

- search box
  - adding to your home page **101**
  - limiting collections displayed in **102**
- search form page, creating a non-tables version **108**
- search results
  - changing document summaries **111**
  - changing document titles in **110**
  - changing information stored in **108**
  - changing the date sort **110**
- search results pages, building **30**
- searches, customizing
  - by collection **122**
  - by description **121**
  - by doctype **122**
  - by document title **119**
  - by form variables **52**
  - by image **120**
  - by imagelink **120**
  - by language **122**
  - by link **120**
  - by meta searches **121**
  - by site **120**
  - by topic **123**
  - by URL **119**
- site
  - customizing searches by **120**
- sitelist.txt file **110**
- skeletons, in patches.py file **88**
- SOAP
  - procedures **97**

- style sheets **82**

## ■ T

- tables
  - creating a non-tables version **108**
  - in Inktomi Search Server **108**
- thesaurus.html **38**
- tips file
  - changing **109**
- topic, customizing searches by **123**

## ■ U

- URL
  - adding to index **110**
  - adding to index automatically **110**
  - customizing searches by **119**

